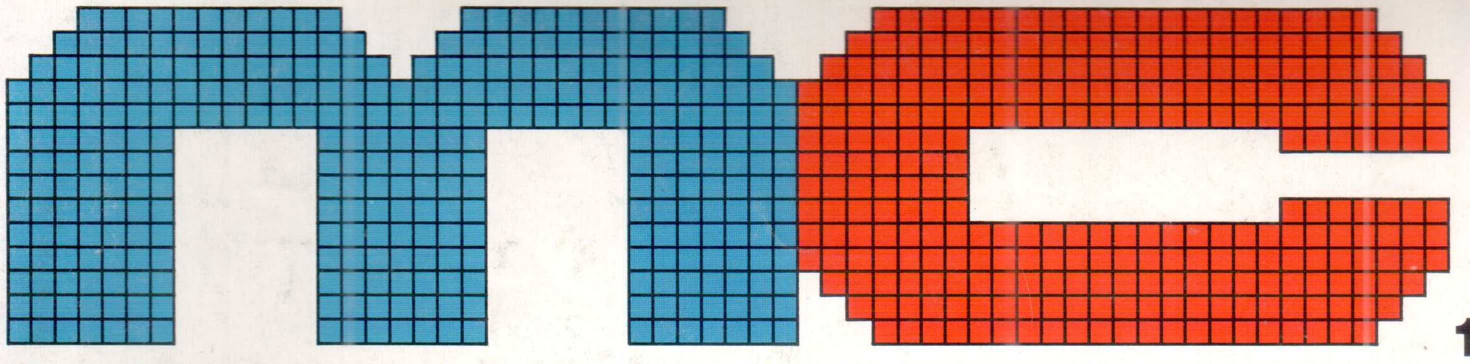


1

Die Mikrocomputer-Zeitschrift

6 DM · 50 öS · 6,80 sfr. · Januar 1982

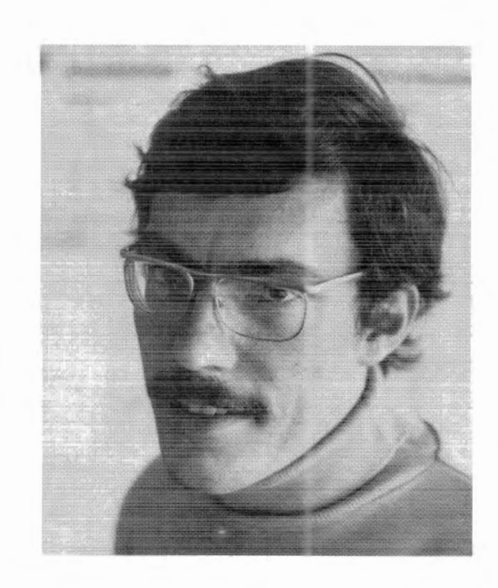
AIM-65/40-Testbericht

Mikrocomputer-Betriebssysteme

Z80-Einplatinen-Computer

Der geknackte CBM

422 neue Z80-Befehle



Standardlösungen

Eine Unmenge von nützlichen Anwendungen wurde inzwischen mit dem mc-Einplatinencomputer „EMUF" realisiert – teils haben wir sie schon veröffentlicht, teils ist das noch vorgesehen, teils sind sie zum Abdruck zu speziell, zu anwenderspezifisch.
Der revolutionär geringe Preis war wohl der Schlüssel zum EMUF-Erfolg. Erkauft wurde er durch die Beschränkung auf ein wirkliches Minimalsystem. Nun sind leider manchmal 16 I/O-Leitungen, ein Timer, 128 Byte RAM und 1 KByte Programmspeicher doch zuwenig; aus diesem Grunde werden wir demnächst einen etwas aufwendigeren Einplatinen-Computer, diesmal mit der CPU 6502 und zwei VIA-Bausteinen 6522, vorstellen.
Und dann gibt es ja noch jene Anwender, die keine Möglichkeit haben, 6502-Programme zu entwickeln und zu testen. In Anbetracht der Tatsache, daß der heutige Tischcomputer-Markt im wesentlichen von den CPU-Typen 6502 und Z80 beherrscht wird, bringen wir deshalb in diesem Heft eine Z80-Lösung. Sie erlaubt die Programmentwicklung mit solchen Computern wie TRS-80, Nascom-1/2, ABC-80, MZ-80K, aber auch mit Computern, die entweder einen Z80-Crossassembler oder eine Z80-Zusatzkarte besitzen (z. B. Apple-II). Bedauerlicherweise haben aber die Entwickler der meisten Z80-Tischcomputer offenbar eine andere Ideologie verfolgt als ihre 6502-Kollegen: Selten findet man bei den Z80-Maschinen solche Dinge wie programmierbare Timer oder I/O-Ports, und auch die interne Interrupt-Organisation entspricht nicht immer den Bedürfnissen von Leuten, die den Tischrechner zur Entwicklung von Programmen für Einplatinen-Computer verwenden wollen. Um z. B. ein solches Programm mit dem TRS-80 zu testen, ist es notwendig, den Computer mit einer externen I/O-Erweiterung auszustatten, da sonst keinerlei Ein- und Ausgabemöglichkeit vorhanden ist. Doch läßt sich diese kleine Hürde leicht durch den Anschluß eines PIA-Bausteins an den Systembus überwinden. Ähnliches gilt für den MZ-80K.
Übrigens läßt sich unser Z80-Einplatinencomputer auch zum vollwertigen Tischcomputer ausbauen – indem man ihn z. B. mit der dynamischen 64-KByte-RAM-Karte aus Heft 4/1981, einer ASCII-Tastatur und einem Video-Interface kombiniert und ein kleines Monitor-Programm ins EPROM schreibt. Zusammen mit einer (schon geplanten) Floppy-Controllerkarte läßt sich sogar ein CP/M-System aufbauen.
Das Schöne an den Einplatinen-Computern ist ja gerade, daß ein und dieselbe Zentraleinheit als Hardware für unterschiedlichste Anwendungen geeignet ist – eine echte Standardlösung also. Was das „Mädchen für alles" letztendlich genau tut, bestimmt ausschließlich jener Tausendfüßler in Form eines EPROM, dessen Programmierung deutlich weniger Schweiß kostet als der Aufbau einer reinen Hardware-Lösung mit einer Europakarte voller TTL- oder CMOS-ICs.
Und es macht geradezu Spaß, „dummen" Geräten durch Einbau eines Einplatinen-Computers eine Art Intelligenz zu verleihen!

Ihr
Herwig Feichtinger

Genie I

Die leistungsstarken '82er Modelle sind da.



Sie sind da. Die '82er Modelle im Geniesystem. Noch leistungsstärker, noch ausgereifter, noch zuverlässiger. Genie I ist die Weiterentwicklung des Video-Genie. Serienmäßig haben alle '82er Modelle Groß-/Kleinschrift mit Unterlängen – incl. deutscher Umlaute. Der ROM-Bereich wurde um 2 K auf 14 K erweitert. Der 12-K-TRS-Level-II-kompatible Bereich ist um viele wertvolle Routinen ergänzt worden. Zum Beispiel: Volle Cursorsteuerung. Außerdem ist durch einen eingebauten NF-Verstärker und -Lautsprecher direkte Tonwiedergabe möglich.

Technische Daten:

- CPU-Z80-Mikroprozessor
- 12 K ROM Microsoft Basic-Software compatibel zum TRS-L II
- Zusätzlich 2 K ROM mit wertvollen Routinen
- 16 K freier Benutzerspeicher – mit Expander erweiterbar auf 48 K und mehr
- Professionelle Schreibmaschinentastatur (ASCII) mit Groß-/Kleinschrift
- Eingebauter Datenrecorder (und Anschluß für zweiten Recorder)
- HF-Modulator für Fernsehanschluß und Monitorausgang für BAS-Norm
- Bildschirmformat 64 x 16 oder umschaltbar 32 x 16 Zeichen, Graphikauflösung 128 x 48 Punkte
- Eingebauter NF-Verstärker und -Lautsprecher
- Systembus auf Edgestecker herausgeführt

TROMMESCHLÄGER COMPUTER GMBH
Flugplatz Bonn-Hangelar · Postfach 2105
5205 St. Augustin 2 · Tel. 02241/20061-62

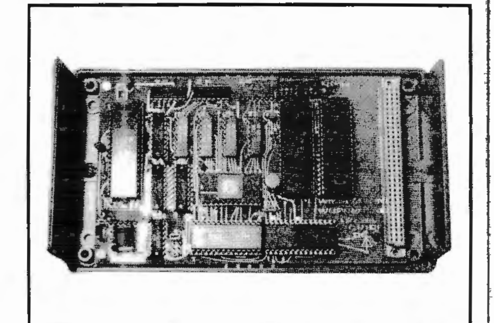
Alleinvertrieb der EACA-Produkte – Unsere Auslandsvertretungen:
Dänemark: DCA, Storegade 39, 6440 Augustenborg 2
Niederlande: Micro Dynamics, Piazza 305, 5611 AG Eindhoven
Österreich: Handelskommerz Wien, Marokkanergasse 3-1-21, A-1050 Wien 3



Test

Wer einen Rechner einsetzen will, benötigt Informationen sowohl über die Hardware als auch über die Software. mc hat deshalb den neuen AIM getestet – ein Ding mit Pfiff für Freunde offener Platinen. Ein weiterer Test betrifft das S-100-System CS 2000, das unter CP/M läuft. Entscheidend beeinflußt wird das Verhalten eines Computers von der Systemsoftware. Zwei große Betriebssysteme stellen wir in diesem Heft vor: CP/M und Unix.

Seite 24 und Seite 76



Einplatinencomputer

In mc sollen Sie jedesmal einen Hardware-Leckerbissen finden. Unser EMUF und die 64-KByte-Speicherplatine sind Beispiele. Ökonomisch sollen die Vorschläge sein und professionell verwendbar. Der Einplatinencomputer mit Z-80 von Reinhard Wiesemann, der in diesem Heft vorgestellt wird, erfüllt diese Bedingungen gut.

Seite 71

MZ-80K-Innenleben

Bezüglich des offenbarten Innenlebens des Sharp MZ-80 K (mc 1981, Heft 4) möchte ich einige ergänzende Bemerkungen machen. Da Sharps Verschleierungstechnik hauptsächlich softwaremäßiger Natur ist, macht eine Entschleierung wenig Probleme. Das Basic SP-5025 wird lesbar nach POKE 10167,1. Am besten führt man diesen POKE-Befehl im Direkt-Modus durch, gibt danach USR (33): USR (36) ein – ebenfalls im Direkt-Modus – und verfährt dann gemäß den Anweisungen, die auf dem Bildschirm erscheinen, jedoch nicht ohne zuvor eine Leerkassette einzulegen. So schafft man sich eine PEEKbare Basic-Version für den privaten Gebrauch, die man nicht immer vorher „freizuPOKEn" braucht.

In Tabelle 3 (S. 41) vermisse ich die Datei-Codes für Assembler-Source-Files (04) und Relocatable Binary Files (05). Ich finde es im übrigen bedauerlich, daß sich Herr Betke bei der Namensgebung der Monitor-Subroutinen nicht an die Sharp-Konventionen gehalten hat. So etwas führt doch nur zu unnötigen Verwirrungen.

Übrigens, Forth für den MZ80K ist erhältlich bei Knight's Computers, 108 Rosemount Place, Aberdeen AB2 4YW, Scotland. Dort bekommt man u. a. auch einen Machine-Language-Loader ohne die Restriktionen der Sharp-Version.

Edmund Ramm, Kaltenkirchen

Sehr gut gefallen

Nunmehr liegt Heft 3/1981 vor mir, und ich muß sagen, daß es mir gut gefallen hat. Daß Sie keine Programme mehr für programmierbare Taschenrechner veröffentlichen wollen, akzeptiere ich.

Ich bin Besitzer eines 64-KByte-Apple und würde mich sehr freuen, wenn Sie doch einmal von den gängigsten Schaltkreisen eine Beschreibung geben würden. Bei Ihrem Druckertest fiel mir auf, daß Sie nicht den wesentlich besseren Drucker MX 82 beschrieben haben, das sollte man vielleicht nachträglich noch tun.

Weiter so, und vielen Dank für „Datenspeicherung mit Videorecordern", was mich am meisten interessiert hat.

Andreas Roemer, Frankfurt/Main

Wir planen keineswegs, programmierbare Taschenrechner ganz aus mc zu verbannen. Viele Probleme sind mit diesen Geräten aber gerade so einfach zu lösen, daß es unsinnig wäre, das fertige Programm abzudrucken, weil es oft nur um das Abtippen einer komplexen Formel geht. Langzeiterfahrungen mit dem neuen MX-82 liegen uns noch nicht vor; wesentliche Eigenschaften, die wir in unserem MX-80-Test feststellten, lassen sich für ihn aber übernehmen. *Die Red.*

Zu wenig Praktikumsstellen

Ich bin 20 Jahre alt und studiere zur Zeit an der FH Augsburg das Fach Informatik. Deshalb möchte ich auch gleich als Betroffener einen Kommentar zur Ihrer Kolumne in mc 1981, Heft 2, loswerden: Es genügt beileibe nicht, wenn man an den Hochschulen und Universitäten den Studiengang Informatik anbietet! Auch die Industrie, die so sehr nach Informatikern verlangt, muß ihren Teil dazu beitragen, denn nach zwei erfolgreichen Semestern darf der fleißige Student sich dann nämlich um einen Praktikumsplatz bemühen. Und plötzlich ist man nirgendwo mehr an Informatikern interessiert. Ich spreche hier aus Erfahrung, denn nachdem im Oktober '80 erstmals Informatik in Augsburg angeboten wurde, ist es jetzt soweit: Am 1. Oktober beginnt das erste Praxissemester. Bei der Suche nach einem Praktikantenplatz zeigten sich die Firmen wenig kooperativ; meines Wissens war nur eine einzige Firma dabei, die sich von vorneherein freiwillig dazu bereiterklärt hat, einige Praktiumsstellen anzubieten. Inzwischen haben wir (die Informatikstudenten) alle einen Praktikumsplatz bekommen, zum Teil mit Hilfe des Praktikantenamtes der FH. Doch wie wird es in einem Jahr werden, wenn für uns das 2. Praxissemester beginnt und für die, die jetzt im Oktober anfangen, das 1. Praxissemester bevorsteht? Dann werden allein in Augsburg voraussichtlich rund 70 Praktikumsplätze benötigt.

Rainer König, Augsburg

X-80SP-Testbericht

Am Max-Planck-Institut steht mir seit einem halben Jahr ein Adcomp-Printer-Plotter X-80SP zur Verfügung, den ich an einem mit GWK-Platinen erweiterten PC-100 betreibe. Dabei habe ich folgende Mängel entdeckt: Es fehlt eine Papierführung für das schon bedruckte Papier. Dieses wickelt sich beim Plotten (Vor-Rückwärtsbetrieb) leicht um die Walze. Das Entfernen einer Papierseite aus dem Drucker erfordert wegen des Fehlens einer FF-Taste (Form Feed) eine umständliche Bedienung des Tastenklaviers. Möchte man innerhalb eines Plottes digitalisieren und den Druckkopf dabei manuell unter Benutzung der Tasten verschieben, so machen die Plot-Koordinaten die Papierverschiebung nicht mit, und man kann anschließend nicht im selben Koordinaten-System weiterplotten. In der ersten Version des Manuals zur RS-232-Version ist der Befehl „Zn m" nicht beschrieben. Obwohl ich dies Adcomp schriftlich mitteilte und einige Monate später ein neues Manual erhielt, fehlt der Befehl auch in diesem.

Klaus Krämer, Stuttgart

Schneller durch Vorsortieren

Eine kleine Ergänzung zu Ihrer Serie über die Sortier- und Suchverfahren:

Lineare Einfügung kostet keine Programmier-, aber viel Rechenzeit. Quick-Sort sortiert schnell, kostet aber nennenswerte Programmierarbeit und -zeit. Lineare Einfügung kostet nur dann viel Rechenzeit, wenn ein Element der Datenmenge die ganze Reihe durchwandern muß. Durch Vorsortieren läßt sich dieser Nachteil beheben. Mit der im folgenden Programm berücksichtigten kleinen Erweiterung der linearen Ergänzung kommt man für Mengen von einigen hundert Daten zu annehmbaren Rechenzeiten. Das Programm ist so simpel, daß es sich selber erklärt (SWAP vertauscht zwei Variablen).

Heinrich Kraft, Kiel

```
10  INPUT „Menge";M:DIM A(M) : B=M
20  FOR I = 1 TO M:A(I) = INT(M*RND) +1 : NEXT
30  GOSUB 170
40  PRINT CHR$(7)
60  B=INT(B/3+.4)
70  ORD=0 :FOR I = 1 TO M-B
80  IF A(I)>A(I+B) THEN SWAP A(I), A(I+B) : ORD=1
90  NEXT
100 IF ORD=1 THEN 70
110 IF B>1 THEN 60
130 PRINT CHR$(7)
140 GOSUB 170
150 END
170 PRINT:FOR W = 1 TO M:
    PRINT USING „# # #";A(W);:NEXT:RETURN
```

Systems '81 von Mikros geprägt

Der Trend, statt Minicomputer lieber die immer leistungsfähiger werdenden Mikrocomputer einzusetzen und damit auch größere EDV-Anlagen zu dezentralisieren, wurde auf der Münchner Systems recht deutlich. In unserer Rubrik „mc-markt" geben wir einen kleinen Ausschnitt aus dem Produkt-Angebot, das auf diesem Sektor mittlerweile fast unübersehbar geworden ist. Mit 660 Ausstellern auf ca. 55 000 m^2 Brutto-Ausstellungsfläche und rund 44 000 Fachbesuchern (davon 3100 auch Teilnehmer des Kongresses „Computersysteme") kann die Münchner Messegesellschaft zufrieden sein. Gegenüber der Systems '79 waren das nämlich 30 % mehr Aussteller, 30 % mehr Fläche und 42 % mehr Besucher.
Schwerpunkte der Aussteller waren Klein- und Mikrocomputer, Computer Aided Design (CAD) und Datenkommunikation. Mit ihr gekoppelt waren fünf Symposien, zehn Benutzergruppen-Seminare und vier Fachseminare. Der Termin für die nächste Systems steht bereits fest: 17. bis 21. Oktober 1983 auf dem Münchner Messegelände.

mc-Strichcode wird ausgebaut

Viel Furore hat der mc-Strichcode gemacht – aber, so argumentierten viele Leser, eigentlich sei er ja nur bei längeren Programmen, speziell bei Maschinenprogrammen interessant. Kurze Basic-Programme kann man ja schnell auch von Hand eingeben.
Richtig! Die kurzen Strichcode-Basic-Programme, die wir bisher veröffentlichten, dienten auch nur dazu, um den Strichcode erst einmal richtig bekannt zu machen. In Zukunft werden wir auch längere Programme im Strichcode bringen; beachten Sie dazu bitte die Hinweise in den nächsten Heften.
Gleichzeitig werden Sie den Strichcode nicht mehr im Heft selbst finden; es wäre sinnlos, -zig Seiten mit schwarzen Strichen zu füllen, mit denen viele Leser gar nichts anfangen können. Also richten wir einen Strichcode-Versand ein: Binnen weniger Tage erhalten Sie auf Anforderung vom Verlag gegen eine geringe Schutzgebühr die zu diesem Heft gehörenden Strichcode-Listings. In Kürze also noch mehr mc-Service – etwas Geduld lohnt sich!

Osborne 1 auf der Systems



Zweckmäßig und kompakt ist der Osborne 1

Der Osborne-Computer, auf der NCC im Frühjahr bereits mit einigem Aufsehen vorgestellt, hat nun auch seinen Weg nach Europa gefunden. Auf der Systems wurde er recht überraschend von der mm-computer GmbH präsentiert. Dieser Rechner ist nicht nur durch seine Hardware von besonderem Interesse, sondern auch durch die Tatsache, daß eine recht umfangreiche Software mitgeliefert wird.
Die Hardware besteht aus einem Z-80-System mit 64 KByte Arbeitsspeicher, zwei Minifloppies mit je 100 KByte, 5-Zoll-Bildschirm, Tastatur mit Zehnerblock sowie IEC- und V-24-Schnittstelle. An Software werden CP/M, CBASIC-2, MBASIC, WORDSTAR und SUPERCALC mitgeliefert.
Das Gerät ist mit all diesen Attributen sehr kompakt, was allerdings nur auf Kosten der Bildschirmgröße möglich ist. Für den, dem der eingebaute Bildschirm zu klein ist, ist der Anschluß eines externen Monitors möglich. Sozusagen das Tüpfelchen auf dem i ist der Preis: unter 5400 DM (ohne MwSt.).

Siemens trennt sich vom PC-100

Während Siemens sich bei seinem Personal-Computer PC-1000 noch vor Beginn der Serienfertigung zur Projektaufgabe entschloß, geht man beim PC-100 (baugleich mit dem AIM-65 von Rockwell) einen anderen Weg: Siemens selbst läßt das Gerät auslaufen; es soll aber mit aller Hardware- und Software-Unterstützung im Lauf des Jahres von einer anderen Firma übernommen werden. Dies betrifft auch die kürzlich fertiggewordene Erweiterung PC-102 mit 12-K-Basic und Mini-Kassetten-Laufwerk, den 8085-Crossassembler und andere Software.
Ganz offensichtlich ist dies – ebenso wie beim PC-1000 – ein innerbetriebliches Kompetenzproblem zwischen dem Bauelemente-Bereich, der sich bisher dieser Mikrocomputer annahm, und der Mittleren Datentechnik, die dieses Treiben immer schon mit Mißtrauen verfolgte. Wenngleich Siemens schon Studien über den Personal-Computer-Markt bei einem Beratungsunternehmen einholte, ist noch nicht abzusehen, ob und wann es Goliath mit den derzeit vorwiegend marktbeherrschenden Davids wieder aufnehmen wird.



Soll von einer anderen Firma weitervertrieben werden: PC-100 von Siemens

Jetzt haben die Deutschen den frechsten Micro-Computer der Welt: alphatronic Micro

Er ist sicher zu bedienen: durch die deutsche DIN-Schreibmaschinen-Tastatur, den Norm-Zehnerblock mit 4 Grundrechenarten und 6 frei programmierbare Funktionstasten.

Er ist übersichtlich: durch praxisgerechten 12″-EDV-Bildschirm mit 24 Zeilen à 80 Zeichen.

Er ist intelligent: durch 48 KB-Speicher und wahlweise 1 oder 2 Disketten-Laufwerke (160 bis max. 640 KB).

Er hat die Supertechnik Intel 8085 A. Erweiterungsfähig durch Steckkarten im Europaformat.

Er hat die entscheidenden Schnittstellen: 2 seriell, 1 parallel, die als IEC-Bus implementierbar ist.

Er hat optimale Druckersysteme: Nadel-, Typenrad- oder Schreibkerndrucker stehen zur Wahl.

Er kann mehrere Programmiersprachen: BASIC, PASCAL, FORTRAN, ASSEMBLER und COBOL. Die deutschsprachigen Bedienungsanleitungen und Programmier-Handbücher sprechen für sich.

Der Service und Kundendienst ist ideal, denn Fachhändler gibt es in jeder größeren Stadt.

Der frechste Micro-Computer der Welt heißt alphatronic Micro und ist von TRIUMPH-ADLER.



Informationsabruf:

Ich möchte wissen, wie frech der alphatronic Micro in der Praxis ist. Bitte senden Sie mir weitere Informationen.
Am meisten interessiert mich folgende Anwendung:

mc 2.12.81/alpha

Name

Titel/Stellung

Straße Tel.

PLZ/Ort

TRIUMPH-ADLER Aktiengesellschaft
für Büro- und Informationstechnik
Vertrieb Micro-Computer
Witschelstraße 71 · D 8500 Nürnberg
Tel. (09 11) 32 05-1 · Telex 6-26 155

TA TRIUMPH-ADLER

alphatronic Micro - der frechste Micro-Computer der Welt

1. Commodore Software-Grand-Prix

Durchstarten in ein neues Schreiben Sie Programme für VC 20 VolksComputer.

1. Preis: Volkswagen Cabrio Fabrikneu in brasilbraun-metallic, aus der Ära, als jedermann ins Automobilzeitalter einstieg.
2. Preis: Commodore CBM 4001-Anlage Zentraleinheit mit Floppy Disk (Doppellaufwerk) und Matrixdrucker.
3. Preis: 1 Original Vespa-Motorroller Das Freiluftvergnügen für junge und junggebliebene »Computer-Freaks«. Führerscheinklasse 3.
4. Preis: 1 Original Fahrrad-Trelo Ein echtes »Siegerfahrrad«, in dem Sie fortan energiebewußt im Trockenen strampeln können.
5.–20. Preis: VolksComputer Zubehör Zur freien Auswahl jeweils im Wert bis zu DM 500,––.
Sonderpreise: Für die besten Ideen: Zum Energiesparen, eines Jugendlichen, Hardware-Modifikationen, angepaßte CBM-Software und alle Programme, die Commodore zusätzlich zu den prämierten übernimmt.



Commodore hat mit dem VC 20 »VolksComputer« nunmehr endgültig das Computerzeitalter für jedermann eingeläutet. Eigene Microprocessor-Technologie und über eine viertel Million verkaufter Mikrocomputer sind hierfür eine ausgezeichnete Basis. Dieses herausragende Know-how erklärt auch die unglaublich günstigen Preise, hinter denen professionelle Leistung steckt:

Grundversion: An jeden Fernseher anschließbar · CPU 6502 B · 5K RAM · 20K ROM · Betriebssystem und BASIC-Interpreter · 24 Farben · 4 Tongeneratoren · Große Schreibmaschinen-Tastatur · Zusätzlich 4 Programmiertasten mit Doppelfunktionen · Steckmodul-Technik · Commodore-BASIC · Math. und Log. Funktionen · **Erweiterung und Zubehör:** Erweiterungsmodul 3K RAM DM 109,––*) · 8K RAM, 16K RAM und Modulbox für 6 Steckmodule in Vorbereitung · ROM auf 24K erweiterbar · IEEE 488 für CBM-Peripherie · RS 232 C Terminal und TTY · Maschinensprache und Assembler über Programmierhilfe-Modul · Floppy Disk (m. DOS) 170KByte ca. DM 1.500–*) · Datassette mit Zählwerk DM 225,––*) · Normalpapier-Traktordrucker, 80 Zeichen breit, DM 1.198,––*) · Hochauflösende Grafik.

*) Jeweils unverbindliche Preisempfehlung.

Der VC 20 ist also nicht nur ein Computer, sondern ein ausbaufähiges und deshalb überlegenes System.


Computer-Zeitalter.
den
1. Preis
Volkswagen
Cabrio
VC 20
Professional Partners
»Startberechtigung« zum 1. Commodore
Software-Grand-Prix anfordern
Wettbewerbs-Regeln (Auszug)
Es dürfen Einzelpersonen oder Gruppen an den Start gehen. Die Programme können auf dem VC 20 VolksComputer oder jedem anderen Computer entwickelt werden. Allerdings ist ein Programmumfang von maximal 11,5 KByte RAM vorgegeben. Die Software muß auf dem VC 20 Volks-Computer in Verbindung mit jedem beliebigen Fernseher laufen. Die Fach-Jury erwartet die Programme auf handelsüblichen Compact-Cassetten mit ausführlicher Beschreibung. Die kompletten Bedingungen, die jedem die gleichen Teilnahmechancen sichern, fordern Sie bitte mit dem Coupon an. Einsendeschluß für die Programme ist der 31. März 1982. Viel Spaß beim Mitmachen!
Bitte senden Sie mir ausführliche Teilnahme-Bedingungen:
Name:
Anschrift:
Einsenden an: Commodore GmbH · Abt. MZ1 · Postf. 426
6078 Neu-Isenburg
DM 899,–
(unverbindliche Preisempfehlung für VC 20 Zentraleinheit.)
Ab sofort beim Commodore-Vertragshandel, Bürofachhandel, in führenden Warenhäusern, guten Rundfunk- und Fernsehfachgeschäften und beim Großversandhaus Quelle.
commodore
COMPUTER

Nicht nur Hobby auf der Hobby-Elektronik

Auf der Stuttgarter Ausstellung für praktische Elektronik und Mikrocomputer, Hobbyelektronik '81, fand man keineswegs nur Spielzeug: Auch Computer mit CP/M-Betriebssystem oder anderen Qualifikationen, durchaus nicht unbefriedigende Peripherie und hochkommerzielle Software konnte man bestaunen, zum Teil auch gleich mitnehmen. Eine Podiumsdiskussion mit Vertretern von Tischcomputer-Herstellern (Sharp, Texas Instruments, Commodore) und einem Versender (Vobis) machte deutlich, welche merkwürdigen Vorstellungen mancherorts noch über die notwendige Dokumentation von Computer-Innereien vorherrschen und welche Probleme es noch mit Software-Vertrieb und Service gibt – einige Hersteller haben es vorgezogen, gar nicht erst zu erscheinen.

Auf dem Mikrocomputer-Stand des Aktionszentrums konnten sich die Besucher firmenneutral von Mitarbeitern der mc-Redaktion beraten lassen. Und auf der Hobby-Börse, einem Flohmarkt, konnte man preisgünstige ASCII-Tastaturen, Computerplatinen oder EPROMs mit nützlichen Programmen erstehen.

Mit rund 30 000 hat sich die Besucherzahl gegenüber dem Vorjahr nur unwesentlich erhöht; die Ausstellerzahl (86) dagegen ist etwas zurückgegangen – vielleicht eine Folge der unüberlegten Terminwahl parallel zur Systems '81 in München.

Betriebssystem für CBM 8096

Der in seinem Konzept etwas verunglückte CBM 8096 hat ein neues Betriebssystem bekommen. Commodore hat das komplette Software-Paket bei der Softwareverbund Mikrocomputer GmbH in München gekauft. Das Betriebssystem stellt dem Anwender 32 KByte reinen Programm- und 32 KByte Variablenspeicher zur Verfügung. Freier Speicherplatz im Bereich des Betriebssystems kann wahlweise dem Programm- oder Variablenbereich zugeschlagen werden.

Das Betriebssystem ist von allerlei Unarten der bisherigen Betriebssysteme von Commodore-Rechnern befreit und um eine ganze Reihe nützlicher Dinge erweitert worden, unter anderem ist der Inhalt des bekannten SM-Kits integriert. Der Betrieb des Rechners mit Floppy-Disk ist Voraussetzung, auf Handling von Kassettenspeichern wurde ganz verzichtet. Weiterhin ist die Zeitdauer für Interruptbehandlung etwa um den Faktor 10 verkürzt worden.

Es ist eine erfreuliche Tatsache, wenn eine Firma wie Commodore die Software für eines ihrer Produkte in Deutschland kauft. Vielleicht macht das Beispiel Schule.

ROM-Listing für TRS-80-Level II

Wer sich in seinem TRS-80 Level II noch nicht genau auskennt, der kann dies mit Hilfe eines 128seitigen ROM-Listings nachholen. Das Listing enthält den vollständig disassemblierten und kommentierten Basic-Interpreter des Rechners sowie über 150 ausführlich dokumentierte Unterprogramme, vom Interpreter belegte RAM-Adressen und eine zusätzliche Auflistung verschiedener Programmsegmente neuerer Versionen.

Diese Fundgrube für alle TRS-80-Benutzer ist bei Luidger Röckrath (Kaiserstr. 54, 4050 Mönchengladbach) zum Preis von 65 DM erhältlich.

ITT-3030: Modular, CP/M und ÄÖÜ

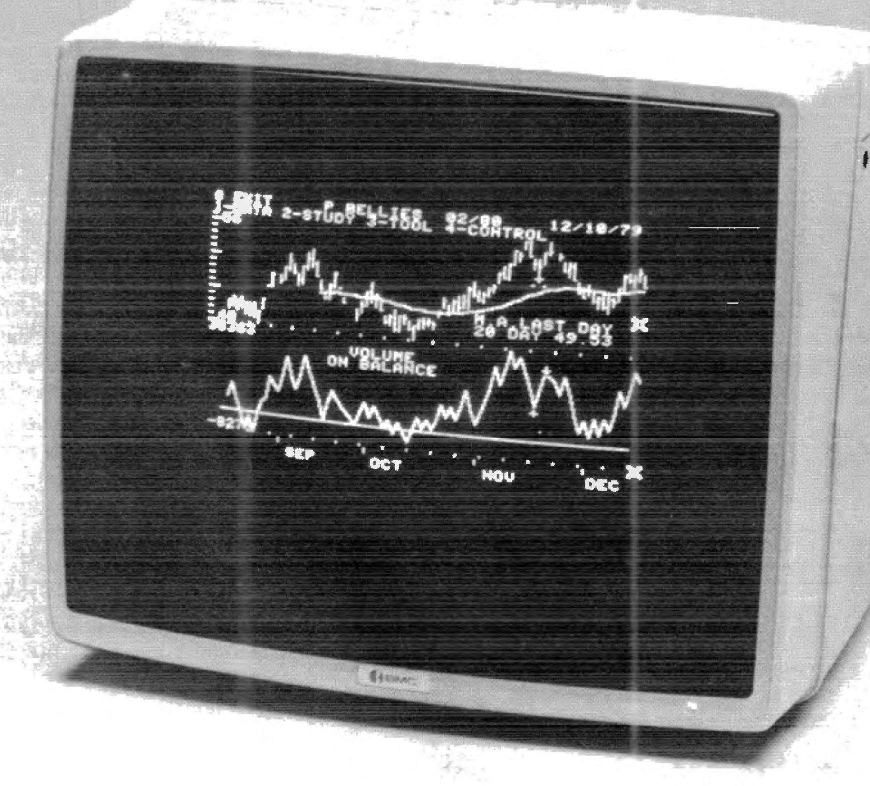


Viel Software dank Standard-Betriebssystemen bietet der Tischcomputer ITT-3030

Keineswegs als Ersatz für den Apple-Nachbau ITT-2020, sondern als Ergänzung nach oben betrachtet ITT den zur Systems '81 neu vorgestellten ITT-3030. Dieser Tischcomputer arbeitet wahlweise mit der CPU 8080, Z80 oder 8086 und mit den Betriebssystemen CP/M (8 Bit) oder BOS (Business Operating System, 16 Bit). Auf einem angeschlossenen Monitor können 24 × 80 Zeichen oder auch hochauflösende Grafiken dargestellt werden. Als Massenspeicher dienen ein oder zwei eingebaute Floppy-Laufwerke (*Bild*). Je nach Ausstattung soll der 3030 zwischen 3500 und 10 000 DM kosten. Und, nicht unwichtig: Er wurde hierzulande entwickelt und besitzt daher auch deutsche Sonderzeichen wie ä, ö, ü und ß. Horst Rosenbaum, Leiter des ITT-Bereiches Audio-Video, der für das Marketing und die Entwicklung der Mikrocomputer bei ITT verantwortlich ist, bezeichnete das Gerät als Baustein innerhalb einer Reihe kommunikationstechnischer Einrichtungen in Büro und Heim, so etwa als Bildschirmtext-Endgerät (wobei man einschränken muß, daß hierfür ja noch keine endgültige Norm besteht!).

Auf die Frage von mc, woran es wohl liege, daß Großunternehmen auf dem Personal-Computer-Sektor anscheinend mehr Schwierigkeiten als ehemalige „Garagenfirmen" haben, räumte Rosenbaum ein, daß größere Firmen bei technischen Neuheiten eher träge reagieren und so unfreiwillig den Kleinen einen technologischen Vorsprung gönnen.

BM 12 EY



Händleranfragen angenehm

Bildröhre	12 Zoll 90°
CRT-Phospor	P4Y (bernsteinfarbig) od. P31 (grün)
Signal	1 V_{ss} an 75 Ω positiv, synchr., neg.
Frequenzen	Hor. 15. 80 kHz, vert. 50 Hz
Bildgröße	21 × 15 cm
Darstellungsformat	1920 Zeichen (80 Zeichen, 24 Zeilen)
Bandbreite	18 MHz (± 3 dB)
Netzanschluß	220 V, 50 Hz, 26 W
Maße	32 br, 27,9 h, 30,8 t in cm
Gewicht	7 kg

Einzigartiger Monitor mit bernsteinfarbiger Röhre



Verkauf über den Fachhandel

General-Vertrieb in Deutschland:
Kleinofen Computer · Kölner Str. 49 · 4000 Düsseldorf 1 · Tel. (02 11) 36 91 91 · Telex 8 582 848

Digitale Tonaufzeichnung

AEG-Telefunken und Mitsubishi (Tokio) werden künftig auf dem Gebiet der digitalen Tonaufzeichnung zusammenarbeiten. Mitsubishi ist Urheber des derzeit dafür verwendeten PCM-Standards, der sich seit Jahren bereits bewährt hat. Telefunken möchte den Standard bei professionellen PCM-Stereo- und Mehrkanal-Magnettongeräten verwenden.
Das von Telefunken/Teldec entwickelte MD-System, die digitale Mini-Schallplatte mit 13,5 cm Durchmesser und 2 × 60 Minuten Abspieldauer, hat offensichtlich das Interesse von CBS, dem größten Schallplattenhersteller der Welt, gefunden. Zwischen den beiden Unternehmen wurden nun weitere Untersuchungen vereinbart.

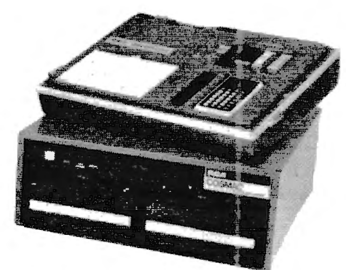
Hannover-Messe: Konzept für VLSI

Unter dem Stichwort „Microtronic" wird das Gebiet elektronische Bauelemente auf der Hannover-Messe bis 1983 neu organisiert. Unter der neuen Überschrift soll eine abgerundete Fachmesse im Rahmen der Hannover-Messe entstehen. Diese neue Teilmesse innerhalb der Messe der Messen wird den Schwierigkeiten der Kommunikation und Information auf dem Gebiet hochintegrierter Schaltkreise und fortschrittlicher hochkomplexer Baugruppen besonders Rechnung tragen. Microtronic wird ein internationales Zentrum anwendungsorientierter Elektronik und Mikroelektronik.
In der neugestalteten Halle 13 soll der Fachbesucher Produkte besichtigen können, Anwendungsfälle studieren können, tiefergehende Informationen anhand eines Schwerpunktthemas (zum Beispiel Elektronik im Kraftfahrzeug) gewinnen können und in einem Kommunikationszentrum an Symposien, Vorträgen und Fachsitzungen teilnehmen können. Wie in einer guten Fachzeitschrift, aber live und hautnah, soll der Besucher die ihn interessierenden Themen aufblättern und bis zum Anwendungsfall auch verfolgen können.
Die Aussteller gruppieren sich mit ihren Ständen um das Zentrum der Halle. Sie können sich auf die eigentlichen Bauelemente konzentrieren, denn die Anwendungsfälle lernt der Besucher schon im Zentrum der Halle exemplarisch kennen. So werden konzentrierte Messegespräche mit informierten Kunden möglich.
Das Konzept wurde mit Vertretern der Industrie (national und international) entwickelt und diskutiert. Man darf auf 1983 wirklich gespannt sein.

288-kBit-Speicherchip von IBM

Dem Entwicklngslabor Burlington der IBM in USA ist es gelungen, einen Versuchsspeicherchip mit einer Kapazität von 288 000 Bits herzustellen. Der neue Chip hat bei nur doppeltem Platzbedarf die 4fache Kapazität wie der erst vor kurzem angekündigte 72-kBit-Chip. Bei dem neuen Speicher handelt es sich um ein dynamisches RAM in der Silizium- und Aluminiumoxid-Halbleitertechnologie (SAMOS) der IBM. Der 288-kBit-Chip wurde in derselben Entwicklungslinie hergestellt wie der 72-kBit-Chip. Es wurden einige voll funktionsfähige Chips produziert, um die Herstellbarkeit zu demonstrieren.
Im Augenblick beabsichtigt die IBM nicht, den 288-kBit-Chip in ihr Produktionsprogramm aufzunehmen.





Das Entwicklungssystem COSMAC IV

Mikroprozessor-familie 1800

Ein Mikroprozessor-Entwicklungssystem namens COSMAC IV wird von RCA für die Prozessorfamilie 1800 angeboten. Das System enthält alle Hard- und Softwareeinrichtungen, die zur Entwicklung nötig sind.
Die Hardware besteht aus der Zentraleinheit mit Bildschirm und Tastatur sowie eingebautem PROM-Programmer und paralleler Drucker-Schnittstelle. Die Software schließt Texteditor, Hilfsprogramme, ein Disketten-Betriebs- und -Verwaltungssystem ein.
Das Ganze wird abgerundet durch einen mobilen Mikromonitor zur Echtzeit-Fehlersuche sowohl für Hardware als auch Software und ein Doppel-Floppy-Disk-Laufwerk mit einer Kapazität von 500 KByte.
Zur erwähnten Mikroprozessor-Familie wird ein neuer Prozessor mit der Bezeichnung CDP1805 angekündigt. Im wesentlichen handelt es sich dabei um einen hinsichtlich Taktfrequenz und Befehlsvorrat verbesserten 1802. Ein schnelles CMOS-ROM CDP1835 (16 KBit) und ein CMOS-RAM MSW5114A (4 KBit) ergänzen das Angebot.
Last not least hat RCA eine Applikationssammlung zum Prozessor 1802 herausgegeben. Das 150seitige Buch in deutscher Sprache soll die Anwendung und Entwicklung von Schaltungen mit dem genannten Prozessor erleichtern. Das Buch trägt die Bezeichnung BMP-801 und ist über RCA-Distributoren erhältlich.

Fünf Jahre Siemens-Intel-Ehe

Nach fünfjähriger Zusammenarbeit auf dem Mikroprozessor-Sektor entwickeln Siemens und Intel nun gemeinsam einen neuen Prozessortyp namens 80286, der eine interne 32-Bit- und eine externe 16-Bit-Architektur aufweist. Er wird voraussichtlich Anfang 1982 lieferbar sein, kann 16 MByte adressieren, besitzt 125 000 Transistoren auf dem Chip und gestattet eine virtuelle Speicherverwaltung für Multi-User-Systeme.
Die Zusammenarbeit von Siemens und Intel wird von Fall zu Fall, also von Produkt zu Produkt entschieden; beim 32-Bit-Prozessor IAPX 432 von Intel gibt es bisher keine Kooperation. Laut Intel ist Siemens in den letzten vier Jahren zu einem gleichwertigen Partner auf technologischem Gebiet geworden; allerdings gab man bei Siemens zu, bisher so gut wie kein Geld mit Mikroprozessor-Produkten verdient zu haben, weil erhebliche Investitionen notwendig gewesen sind. Immerhin erreichte der Umsatz solcher Produkte bei Siemens aber inzwischen 100 Mio. DM, 20 000 Leute wurden durch die Siemens-Mikrocomputer-Schule geschleust, und 20 000 Seiten Dokumentation sind entstanden!

Neuer Stern: Sirius I

Von der erst im Jahre 1981 gegründeten Firma Sirius Systems Technology wurde ein Mikrocomputer vorgestellt, der auf der Systems '81 beträchtliche Aufmerksamkeit erregte: der Sirius I. Er ist mit einer bemerkenswerten Ausstattung versehen: CPU 8088, 128 KByte RAM (erweiterbar bis 512 KByte) und 2 Mini-Diskettenlaufwerke mit zusammen 1 MByte Speicherkapazität. Die Kapazität der Floppy-Disks kann auf 2 MByte erhöht werden, außerdem ist der Anschluß eines Winchester-Laufwerks mit 10 MByte möglich.

Mit der abgesetzten Tastatur und dem dreh- und kippbaren Bildschirm genügt das Gerät auch ergonomischen Gesichtspunkten. Der 12-Zoll-Bildschrim hat eine Auflösung von 800 × 400 Punkten und ermöglicht eine sehr gute grafische Darstellung. Verbindung mit der Außenwelt kann man über eine RS-232-(V-24-) Schnittstelle herstellen oder mit einer programmierbaren Parallelschnittstelle von 8 Bit Breite (Druckeranschluß). Als Betriebssystem findet CP/M-86 von Digital Research Verwendung, das auch in dem neuen Mikrocomputer von IBM eingesetzt wird. Dadurch kann auf ein großes Software-Reservoir zurückgegriffen werden. Auch eine ganze Reihe von höheren Programmiersprachen wird vom Hersteller angeboten.

Das besondere Interesse an diesem Rechner wird natürlich nicht zuletzt durch einen der Firmengründer ausgelöst, dessen Name nicht ganz unbekannt ist: Chuck Peddle, der u. a. die CPU 6502 und den PET entwickelte.



Die Grundkonfiguration des Sirius I mit 128 KByte RAM soll nicht mehr als 12 000 DM kosten

VME-Bus für 16-/32-Bit-Systeme

Von den Firmen Mostek, Motorola und Philips/Signetics ist ein neuer Mikrocomputer-Systembus gemeinsam entwickelt und auf der Systems vorgestellt worden. Dieser Zusammenarbeit liegt natürlich die CPU 68000 zugrunde, die von allen beteiligten Firmen hergestellt wird oder werden soll. Kurzfristig hat sich auch Thomson diesem System angeschlossen.

Als Endergebnis dieser Entwicklung ist der VME-Bus entstanden, der in Mikrocomputer-Systemkarten dieser Firmen zukünftig Verwendung findet, wobei von Motorola auf der Systems bereits die ersten Systemkarten mit VME-Bus-Interface vorgestellt wurden. Das Kartenformat sowie die eingesetzten Steckverbinder entsprechen DIN 41494 bzw. DIN 41612, allgemein bekannt als Europakarte.

Die Entscheidung zugunsten des Europakartenformates fiel nicht von ungefähr, die Verbreitung auf dem europäischen Markt sowie die steigende Akzeptanz dieses Standards auf dem amerikanischen Markt waren ausschlaggebend.

Es ist geplant, den VME-Bus den internationalen Normungsorganisationen vorzulegen.

Systemkarten mit VME-Bus-Interface, hier im Format einer Doppel-Europakarte

Prozessoren mit Basic-Interpreter

Das Programmieren von Mikroprozessoren in maschinennahen Sprachen ist nicht jedermanns Sache. Systeme, die sich in höheren Sprachen – meist in Basic – programmieren lassen, haben den Nachteil, daß sie deutlich teurer und für zeitkritische Anwendungen schlecht brauchbar sind.

Seit diesem Sommer gibt es in USA gleich zwei neue Prozessoren, die in ihrem ROM einen Basic-Interpreter haben. Damit lassen sich einfache Systeme für Steuerung usw. aufbauen und in dem zwar nicht sehr umfangreichen, aber erstaunlich leistungsfähigen „Tiny-Basic" einfach programmieren.

Es handelt sich einmal um den Prozessor INS 8073 von National-Semiconductor; zum anderen um den Z 8671 von Zilog. Für beide Prozessoren gibt es in USA bereits kleine „Systeme" zu kaufen; man darf gespannt sein, wann diese auch hier auftauchen.

Programmieren in Maschinensprache mit dem 6502

Einführung und Programme. Von E. Flögel. 247 Seiten, zahlreiche Abbildungen, kart. 49 DM. Hofacker-Verlag, München.
ISBN 3-921682-61-4

Der im Untertitel versprochenen Einführung zuzurechnen sind die Kapitel Aufbau der CPU, Befehls- und Adressierungsarten. Dieser Teil vermittelt einige Grundkenntnisse, wenngleich die entsprechenden Handbücher der System-Hersteller damit nicht ersetzt werden können.
Anschließend wird auf die Programmentwicklung per Assembler eingegangen, ergänzt durch fast 70 Seiten Programmbeispiele. Leider sind diese Programmbeispiele ausschließlich auf den Apple zugeschnitten und teilweise bar jeden Kommentars. Dadurch wird das Anpassen dieser Programme an andere Rechner erschwert. Das Kapitel über 6502-Systeme, gemeint sind die wichtigsten, heute am Markt befindlichen Mikrocomputer mit 6502-CPU, zeigt anhand von kurzen Beispielen ein paar Eigenheiten der Geräte auf. Hier wäre vielleicht der Hinweis auf Assembler und Assemblierungshilfen für ausgesprochene Basic-Rechner wie CBM nützlich gewesen. Den Abschluß des Buches bilden ein Kapitel über Interface-Bausteine und deren Programmierung sowie eine Befehlsliste des 6502 im Anhang.
Der schon etwas fortgeschrittene Programmierer wird sicher eine Reihe verwertbarer Programme finden, für den Anfänger scheint dieses Buch weniger geeignet. Sn.

Pascal: Einführung, Programmentwicklung, Strukturen

Ein Arbeitsbuch mit zahlreichen Programmen, Übungen und Aufgaben. Von Jürgen Plate und Paul Wittstock. 395 Seiten, 178 Abbildungen. Lwstr-geb. 48 DM. Franzis-Verlag, München.
ISBN 3-7723-6901-4

Dieses Buch gehört zu den wenigen Pascal-Programmieranleitungen, die nicht voraussetzen, daß der Leser vorher schon eine andere Programmiersprache beherrscht. Die Autoren setzen nicht voraus – sie erklären einleitend die Grundzüge der Computer-Hardware, zeigen wie Programme im Computer ablaufen und wie man Probleme in strukturierte Lösungen umsetzt. Schrittweise lernt man die einzelnen Pascal-Schlüsselworte kennen, erfährt, welche Datentypen es gibt und wird auch mit typischen Fehlermeldungen vertraut.
Die im Buch aufgeführten Beispielprogramme sind ausführlich erläutert und wurden auf einem Großrechner getestet und ausgedruckt. Das Manko vieler anderer Programmier-Einführungen, daß zahlreiche Fehler in Beispielprogrammen den Leser oft zur Verzweiflung bringen, dürfte damit wohl wirksam vermieden sein. Fazit: Diese „Pascal-Bibel", auch als Pascal-Fibel zu bezeichnen, ist sowohl für die EDV-Ausbildung als auch als Handbuch für die Praxis ideal geeignet. Fe.

MICRO/Apple

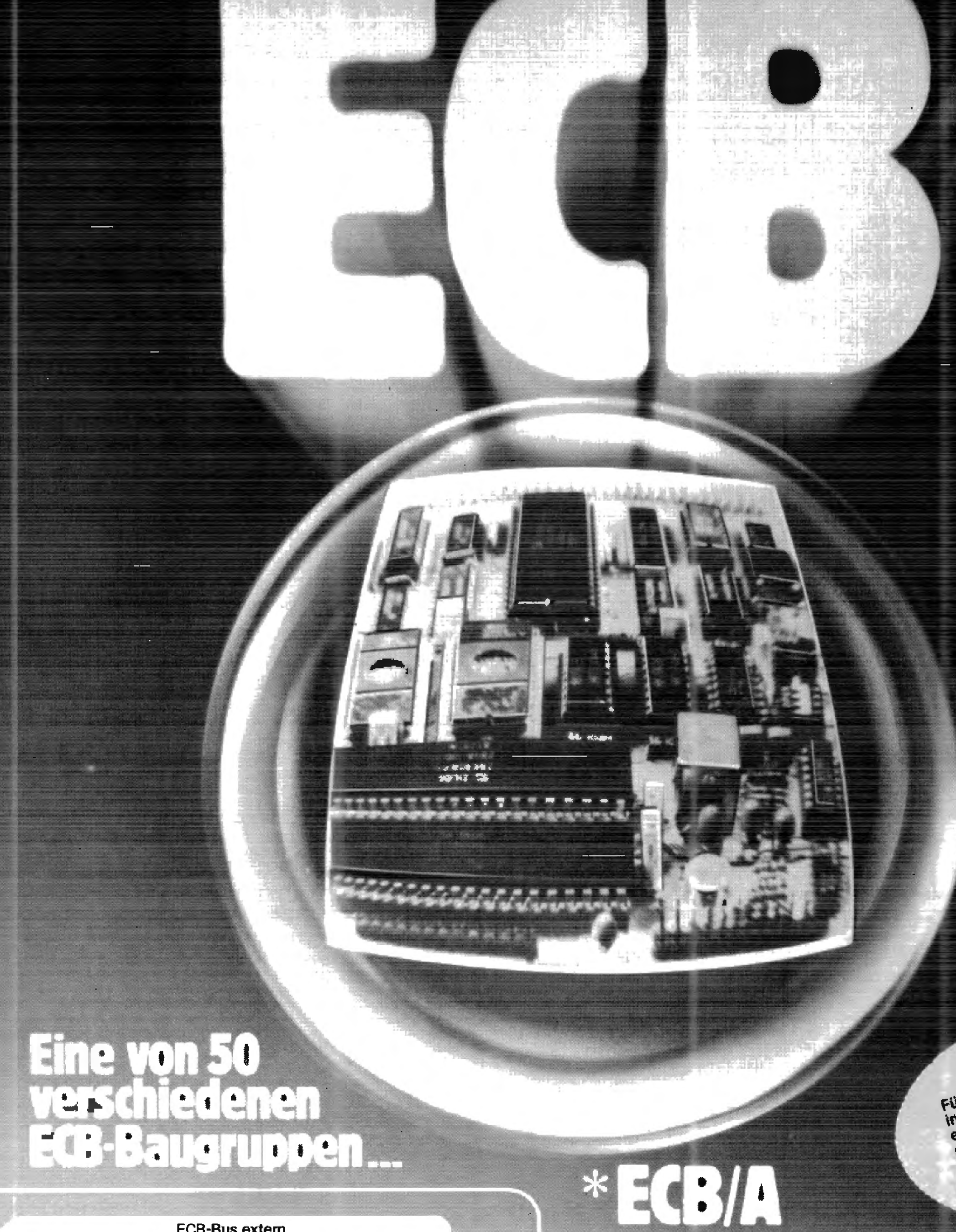
Herausgeber: Ford Cavalari. 216 Seiten, zahlreiche Abbildungen, Tabellen und Listings. 85 DM. In Deutschland erhältlich bei: MSB-Verlag, Markdorf.
ISBN 0-938222-05-8

Das wünscht sich so mancher Computerbesitzer: ein Buch, in dem nur das eigene System behandelt wird. Die vorliegende Zusammenstellung von Artikeln, die in der amerikanischen Zeitschrift MICRO erschienen sind, richtet sich an Benutzer des weitverbreiteten Apple. Beschrieben werden in erster Linie Programme in Applesoft- und Integer-BASIC. Um dem Leser das mühsame Eintippen zu ersparen, liegt dem Buch eine Diskette bei, die unter DOS3.2 erstellt wurde. Darauf befinden sich knapp 40 Programme. Thematisch gliedert sich das Buch in sieben Teile: BASIC-Hilfsprogramme, Ein-/Ausgabe-Verbesserungen, „Utilities" für Anwenderprogramme, Grafik, Lern- und Lehrprogramme, Spiele sowie Wissenswertes zum Nachschlagen. Die Fülle des Materials wird sowohl den „ernsthaften Programmierern" gerecht, als auch denen, die einfach fertige Programme laufen lassen wollen. Als Beispiel für die eine Seite stehen Beiträge über Dienstroutinen wie „Renumber" und „Shape-Table-Generator", auf der anderen Seite findet man das bekannte Simulationsmodell „Life" oder ein Programm zur Berechnung der Planetenbahnen. Den großen Reiz des Buches macht aber weniger der Wert der Software aus, sondern vielmehr die große Zahl der Anregungen und der zweifellos erzielbare Lerneffekt. Ho.

Basic-Interpreter

Funktionsweise und Implementierung in 8080/Z80-Computer. Von Rolf-Dieter Klein. 172 Seiten, 43 Abbildungen. Lwstr-geb. 32 DM. Franzis-Verlag, München.
ISBN 3-7723-6941-3

Nicht jeder Mikrocomputer ist von Haus aus mit einem Basic-Interpreter ausgestattet – sei es, daß er vom Betreiber selbst gebaut wurde, oder daß er zunächst nur für das Programmieren in Assembler konzipiert war. Hier wird nun endlich einmal beschrieben, wie ein Basic-Interpreter überhaupt arbeitet und wie er in einen vorhandenen Computer implementiert werden kann. Der Autor stellt mehrere Interpreter-Programme für die CPU-Typen 8080, Z80 und – ein Leckerbissen! – für die neue 16-Bit-CPU Z8000 vor, zum Teil als gut dokumentiertes Assembler-Listing, zum Teil auch als Hex-Dump mit den nötigen Angaben zur Anpassung der Eingabe- und Ausgaberoutinen. Wer dieses Buch besitzt, lernt nicht nur die Arbeitsweise solcher Interpreter verstehen, sondern wird – wenn er über die Möglichkeit zur Assembler-Programmierung verfügt – auch in die Lage versetzt, selbst einen individuell zugeschnittenen Interpreter für irgendeine Programmiersprache zu erstellen. Denn ob Basic, Pascal oder Forth – die Befehlsdecodierung und Arithmetik bleibt im Grunde immer dieselbe. Fe.



ECB*

mc quickie – die schnelle Produktanzeige!

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.



GWK GESELLSCHAFT FÜR TECHNISCHE ELEKTRONIK mbH.

AIM 65/40

Der große professionelle Kleincomputer für den Einsatz in Forschung, Entwicklung und Fertigung.

- Intelligentes Display, 40stellig
- Intelligenter Drucker, 40stellig
- Terminal-Tastatur mit Groß- und Kleinschreibung
- Adressiert bis zu 128 KByte
- Bis zu 48 KByte RAM on Board
- Bis zu 32 KByte EPROM on Board
- Erweiterung durch GWK-Expansion-System
- GWK Extended Basic

Bitte fordern Sie ausführliche Unterlagen an.

Asternstraße 2, D-5120 Herzogenrath
Telefon (0 24 06) 6 23 94
Telex 8 32 109 gwk d

Finanz-Buchführung

für alle COMMODORE-CBM-Gerätekombinationen.
Kontenplan nach IHK-Empfehlung einprogrammiert.
Nach deutschem Steuerrecht für Einkommen- und Körperschaftssteuer.
Fortschreibung von Januar bis Dezember.
Datensicherung: Kassette und/oder Diskette.

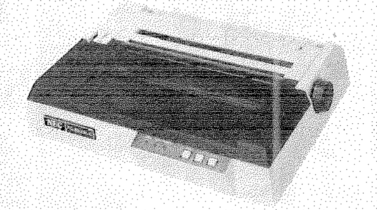
Nach Eingabe der Geschäftsvorgänge jederzeit Journal, Kontoauszüge mit Monats- und Jahreszwischensaldo, Vorsteuerguthaben, Wareneingangsbuch, Zwischenbilanz und GuV-Rechnung.

Jahresabschluß mit Inventurerfassung, Inventurlisten nach Warengruppen getrennt, Handelsbilanz, Steuerbilanz und Jahres-GuV-Rechnung.

Alles schnell, übersichtlich und sicher.
Programmpaket Kfm. Buchführung **DM 1295.–.**
Ausführliches Handbuch **DM 25.–.**
Kurz-Info gegen Freiumschlag A5 (1.50).

FS BAUMGARTEN G.m.b.H.
Juister Weg 11, 3000 Hannover 1
Entwickelt Programme für Kleincomputer-Systeme.

PC-8023B-C



NEC PC 8023B-C – Der neue Standard

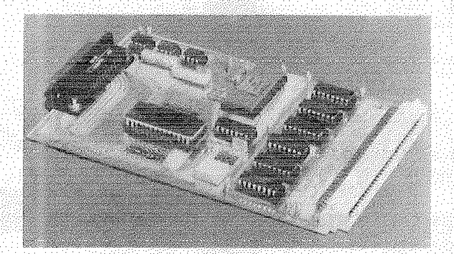
Noch nie gab es so viel Leistung für so wenig Geld!
- 4 Schriftarten: Pica, Elite, Kompress u. Proportional
- Druckgeschwindigkeit 100 Zeich./s
- Hochauflösende Dot-Grafik
- Schnittstellen für fast alle Micros
- zum Superpreis **DM 1950.–** inkl. MwSt.

Microcomputersysteme Ingeborg Strie
Ringstr. 180, 2831 Sudwalde, Tel. (0 42 47) 12 30

Graphik-Software für Z80

Das Softwarepaket **GRAPHIC 80** ermöglicht das Erstellen von Bildern in hochauflösender Graphik von 512 × 256 Punkten. Unterprogramme zum Zeichnen von Vektoren und Punkten sind verfügbar.
Ein Terminalprogramm ermöglicht gemischten Betrieb von Graphik und ASCII-Darstellung mit Scrolling.
GRAPHIC 80 unterstützt den **ELZET-80**-Monitor in Verbindung mit einer **RAM-GRAPHIK**-Karte, wodurch eine herkömmliche **Video**-Karte überflüssig wird.
GRAPHIC 80 ist für **CP/M**-Betriebssystem lieferbar.
Preis: DM 155.– + MwSt. (175.15) zuzügl. Datenträger

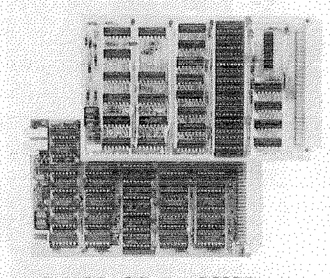
Deutschland:	Schweiz:
SCHECK ELEKTRONIK	**H. BERNHARD ELECTRONIC**
Hockenheimer Str. 58	Aarauer Straße 20
D-6831 Neulußheim	CH-5734 Reinach/AG
Tel. (0 62 05) 3 31 58	Tel. (0 64) 71 69 44



Neu: 16-Kanal-A/D-Wandler 10 Bit

- **ECB-Bus Z80**-kompatibel
- **16** Eingänge mit **10 Bit** Auflösung
- Schnelle Wandlungszeit: 40 µs/Kanal
- Eingänge: 0–5 V, 0–10 V, 0–±5 V, 0–±10 V
- Über Portadresse adressiert
- Anschluß an µP 6800, 6502 möglich
- Treibersoftware (Listing) im Lieferumfang
- Preis: DM 440.– + 13% MwSt. (498.33)
- **Entwicklung spezifischer Hard- und Software**

SCHECK ELEKTRONIK
Hockenheimer Straße 58, D-6831 Neulußheim,
Telefon (0 62 05) 3 31 58



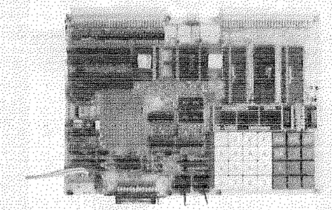
Neu: RAM-GRAPHIK FARBvorbereitet, SW

- **ECB-Bus Z80**-kompatibel
- **512 × 256** Bildpunkte
- Anzusteuern wie **16 K** statisches RAM
- **FARB**- bzw. **GRAUSTUFEN** sind durch einfaches Kaskadieren möglich, z. B. **3** Karten = **8** Farben
- Verschiedene Optionen, z. B. **Card-Select** über eigenes Port auf Platine, nur 5-V-Vers. usw.

Preis: **FARB**vorbereitet DM 648.– + MwSt. (732.24)
SW DM 398.– + MwSt. (449,74)

Deutschland:	Schweiz:
Scheck Elektronik	**H. Bernhard Electronic**
Hockenheimer Straße 58	Aarauer Straße 20
D-6831 Neulußheim	CH-5374 Reinach/AG
Tel. (0 62 05) 3 31 58	Tel. (0 64) 71 69 44

BETA 65



Der erste wirklich universell einsetzbare Single-Board-Computer! BETA 65 ist ein äußerst preisgünstiges System für viele Anwendungen – vom Selbststudium (Lehrsystem) bis zur Prozeßsteuerung:
- mit dem am weitesten verbreiteten Prozessor **6502**
- bis zu **52** I/O-Leitungen
- extrem leistungsfähiger Monitor (4 K)
- Hex-Assembler und -Editor, 2-K-RAM
- Kassetten-Interface und RS-232
- erweiterbar (u. a. mit BASIC)
- preisgünstig: **DM 498.–** inkl. MwSt.

WOLFRAM FEISE
MICROPROZESSORTECHNIK
Alte Zeche 2, D-3013 Barsinghausen 4
Postfach 15, Tel. (0 51 05) 6 29 27



alphaTronic Software

Dental-Labor-Programm	**DM 5645.–**
Anästhesie-Daten-Erfassung	**DM 6775.–**
Adreßverwaltung	**DM 506.–**
Textbearbeitung	**DM 619.–**
Standard-Brief (Text- u. Adreßbearb.)	**DM 1015.–**
Alphatronic-Dienstprogramme	**DM 1015.–**
Finanzbuchhaltung	**auf Anfrage**
Auftragsabwicklung „ROSI"	**auf Anfrage**

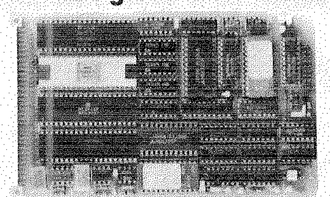
Alle Preise inkl. MwSt.
Händlerpreise auf Anfrage.
Kurzinfo gegen Freiumschlag von:

ES-EM Software U. Siegmund u. J. Miehlke
Brunhildenweg 6, D-5000 KÖLN 90,
Telefon (0 22 03) 6 62 22

mc quickie – die schnelle Produktanzeige!

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

Die intelligente Terminalkarte

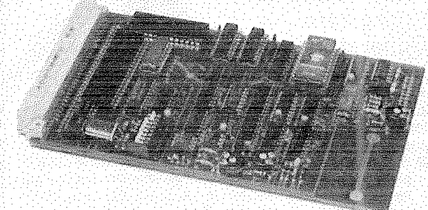


Video 4

- Videosystem mit eigener Z-80-CPU
- Bildformat: 80 Zeichen/Zeile, 24 Zeilen/Bild
- Vielfältige Funktionen, z. B. frei adressierbarer Cursor
- Beliebige Zeichensätze inkl. Grafik, z. B. auch APL
- Attribute: Invertieren, Blinken, Hellschrift u. Farbsteuersignale
- Tastatur u. Druckeranschluß – und vieles mehr

Computer Elektronik GEORG KRAUSE
Zum Römergrund 59, 6501 Wörrstadt
Telefon (0 67 32) 41 78

Die Verbindung vom Rechner zum Bildschirm



Videointerface VIDEO 3

- Serielle Schnittstelle mit 50–19 200 Baud
- Paralleler Tastaturanschluß
- 2 Zeichensätze serienmäßig (ASC II + deutsch)
- Attribute: Blinken, Breitschrift u. Hellschrift
- Option: UHF-Modulator, sep. Sync.-Impulse
- Preis: DM 398.– netto (DM 449.74 inkl. MwSt.)

Computer Elektronik GEORG KRAUSE
Zum Römergrund 59, 6501 Wörrstadt
Telefon (0 67 32) 41 78

FlopCo bietet an:

- **Centronicsdrucker 101AL**
  gebraucht und generalüberholt
  165 Zeichen/Sek. 132 Zeichen/Zeile
  10 Zeichen/Zoll 4 bis 14 Zoll Traktorbreite
  Einzelstückpreis DM 1500,–
- **Qume Sprint 5** Typenraddrucker
  Ausstellungsstücke
  bis 55 Zeichen/Sek. 132 bzw. 158 Zeichen/Zeile
  6 Zeilen/Zoll Traktor- und Andruckwalze
  Standard Qume Schnittstreue
  Einzelstückpreis DM 4000,–
- Centronics kompatible Schnittstelle in Verbindung mit Qume Sprint 5
  Einzelstückpr. DM 250,–/alle Preise inkl. MwSt.

FlopCo Datentechnik
Schlehenhag 15
8068 Pfaffenhofen

SOFTWARE

ECKHARDT UND SCHAAL GMBH

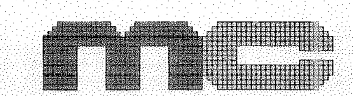
Software für: Commodore-Computer

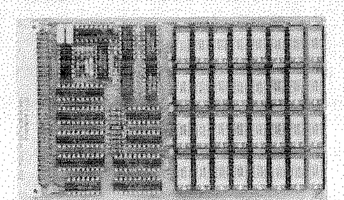
Finanzbuchhaltung
Lagerwirtschaft/Auftragsabwicklung
Karteiverwaltung/Text
Zahlreiche Branchen-Pakete:
z. B. Zahnärzte, Mitgliederverwaltung usw.
Individuelle Anpassungen
BASIC-COMPILER (PETSPEED)
und Questar/M

Bitte nennen Sie uns Ihre Wünsche

ECKHARDT UND SCHAAL GMBH
Zweigertstr. 12, 4300 Essen 1
Tel. 02 01/77 30 53-54

mc 64-KBYTE-RAM-KARTE



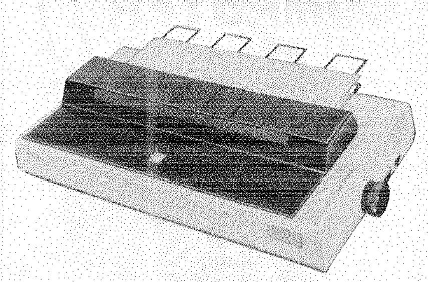


Quasi-statischer Betrieb durch selbständigen Hidden Refresh. Damit für alle 8-Bit-Systeme geeignet. Beliebige Ausblendung v. 4-K-/8-K-Bereichen od. Banking möglich. ECB-Bus-kompatibel, Leistungsaufnahme 3 W max. Systemfrequenz 5 MHz (Opto. 6 MHz) bzw. 1 MHz (1,5 MHz).

Preise für geprüfte Fertigplatinen (Bausätze) inkl. MwSt., bestückt mit:			
	16 KByte	425.–	(225.–)
	32 KByte	517.–	(279.–)
	48 KByte	608.–	(333.–)
	64 KByte	690.–	(395.–)

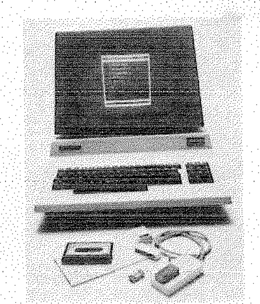
F. Oettle · Stifter Straße 40 · 8902 Neusäß
Telefon (08 21) 48 18 80

EPSON MX-100 F/T



40 cm Papierbreite, 136 Zeichen/Zeile, auf 233 Zeichen/Zeile umschaltbar, bidirektion, wegoptimiert, hochauflösende Grafik, Traktorführung u. Einzelblatteinzug.

DATA-SERVICE GmbH
Computer-Systeme · Software · Zubehör
Autorisierter Vertragshändler mit Kundendienst
6740 Landau/Pf., Kramstr. 23
Tel. (06341) 84577 u. 20729

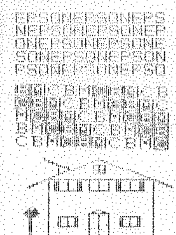


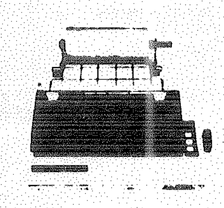
EPROM-Programmiergerät

für 2-K- und 4-K-Einspannungs-EPROMs. Anschlußfertig im Gehäuse für CBM-Serie 3000-8000 inkl. anspruchsvoller Software. Kein Extra-Netzteil erforderlich.
Preis inkl. MwSt. nur **DM 298.–**
EPROM-kompatible 2-K- und 4-K-CMOS-RAMs mit Pufferakku zum Entwickeln von EPROM-Software.
Preis inkl. MwSt. 2K **DM 298.–** 4K **DM 348.–**

U. Schulz Datentechnik, Tel. (0 41 81) 3 65 65
Meilsener Birkenweg 12a, 2110 Buchholz

Görlitz Computerbau liefert : MX 80 Typ II





Das bieten nur wir Ihnen mit diesem Gerät: echte Einzelpunktgrafik, EPSON-Schrift sowie mit ROMs für CBM auch mit CBM-Schrift, also alle CBM-Zeichen, Grafik und RVS! Wir drucken auch Bildschirm-Grafik in High-Resolution (64000 Punkte) mit diesem Drucker aus!

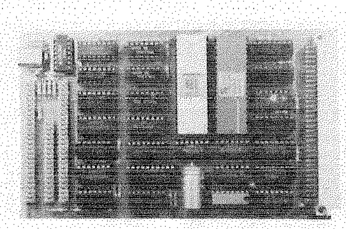
MX 80 Typ II Traktor........................DM 1825,- inkl.
MX 80 Typ II F/T...............................DM 2025,- inkl.
MX 80 Typ II T, jedoch mit CBM-I....DM 1945,- inkl.
MX 80 Typ II F/T mit CBM-I............DM 2145,- inkl.

Die CBM-Versionen enthalten Zusatz-ROMs für Serie CBM 3000, keine weiteren Interfaces notwendig !

UNSERE INTERFACE – OFFENSIVE FÜR CBM :

12-bit 4-Kanal A/D und 2-Kanal D/A-Wandler (1 Platine)..... DM 795,-
Schrittmotor-Interface für 4 Schrittm. vorw./rückw. .... DM 595,-
TTY-, V24- und weitere Interfaces lieferbar. Sonderanfertigungen für alle Zwecke. Alle Interfaces arbeiten voll mit dem Betriebssystem, also kein PEEK, POKE oder SYS erforderlich!

GÖRLITZ COMPUTERBAU POSTFACH 852
5400 KOBLENZ TEL. 0261–27500



Z80-Floppy-Disk-Controller

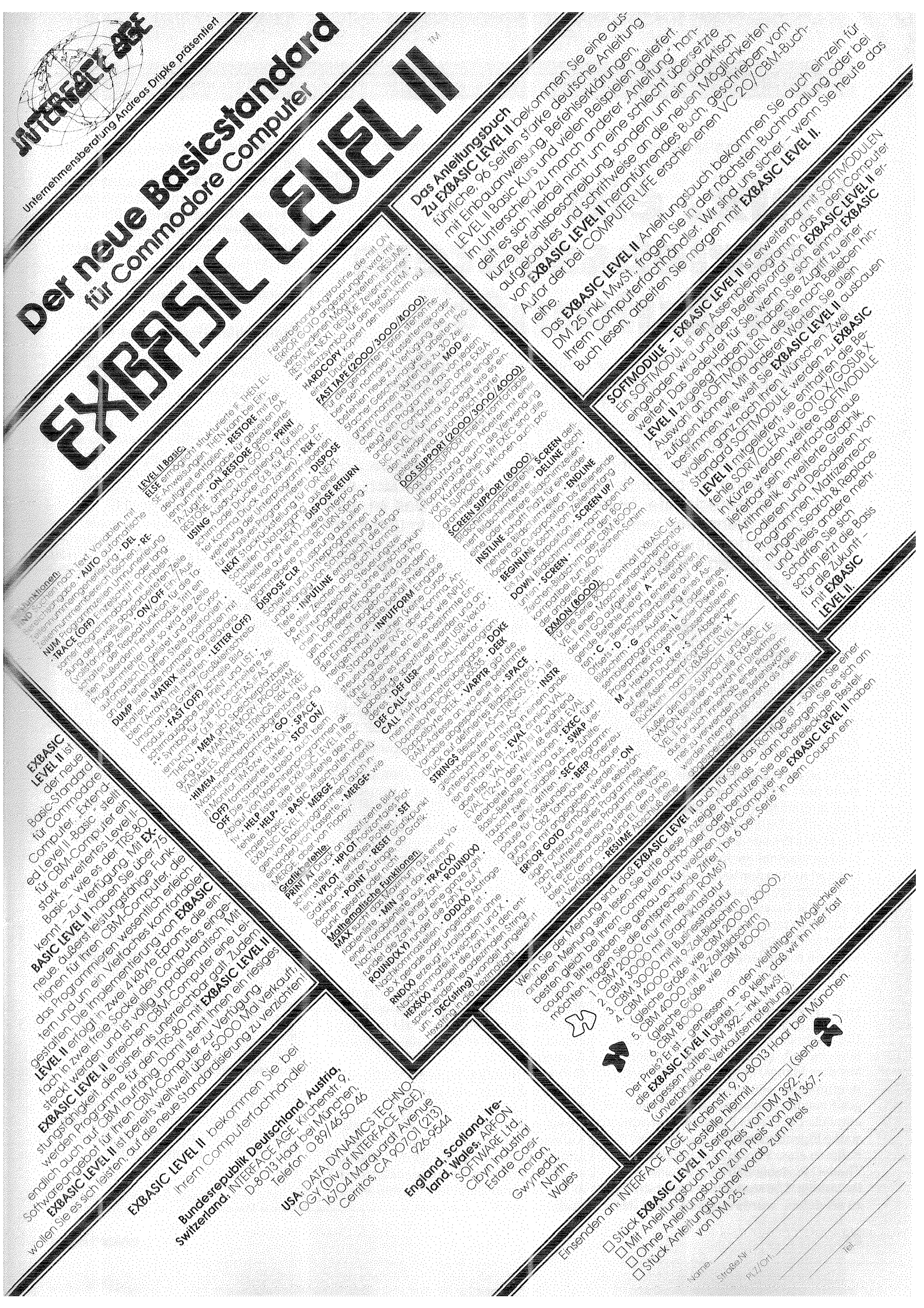
- Europakartenformat mit ECB-Bus-Belegung
- 8"- oder 5,25"-single- oder double-density
- Controller: NEC µPD 765
- Z80-DMA mit BAI/BAO und IEI/IEO-Struktur
- Anschlüsse für Shuggart- oder Phillips-Bus
- Mit Beispielsoftware als Source-Listing

Bestückte und getestete Platine .... **DM 932.25**
Bausatz .... **DM 678.–**
Low-cost-Version (ohne DMA) für BASF 5,25" 559.35 DM (best.) und 372.90 DM (Bausatz)
Alle Preise inkl. Mehrwertsteuer

Janich und Klass Computersysteme
Im Ostersiepen 76, 5600 Wuppertal 1
Telefon (02 02) 43 00 30/42 58 17

mc quickie – die schnelle Produktanzeige!

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.


Unternehmensberatung Andreas Dripke präsentiert
INTERFACE AGE
Der neue Basicstandard für Commodore Computer
EXBASIC LEVEL II ™
Das Anleitungsbuch
Zu EXBASIC LEVEL II bekommen Sie eine ausführliche, 96 Seiten starke deutsche Anleitung mit Einbauanweisung, Befehlserklärungen, LEVEL II Basic Kurs und vielen Beispielen geliefert. Im Unterschied zu manch anderer „Anleitung" handelt es sich hierbei nicht um eine schlecht übersetzte kurze Befehlsbeschreibung, sondern um ein didaktisch aufgebautes und schrittweise an die neuen Möglichkeiten von EXBASIC LEVEL II heranführendes Buch, geschrieben vom Autor der bei COMPUTER LIFE erschienenen VC 20/CBM-Buchreihe.
Das EXBASIC LEVEL II Anleitungsbuch bekommen Sie auch einzeln für DM 25 inkl. MwSt., fragen Sie in der nächsten Buchhandlung oder bei Ihrem Computerfachhändler. Wir sind uns sicher – wenn Sie heute das Buch lesen, arbeiten Sie morgen mit EXBASIC LEVEL II.
SOFTMODULE – EXBASIC LEVEL II ist erweiterbar mit SOFTMODULEN. Ein SOFTMODUL ist ein Assemblerprogramm, das in den Computer eingeladen wird und den Befehlsvorrat von EXBASIC LEVEL II erweitert. Das bedeutet für Sie, wenn Sie sich einmal EXBASIC LEVEL II zugelegt haben, so haben Sie Zugriff zu einer Auswahl an SOFTMODULEN, die Sie nach Belieben hinzufügen können. Mit anderen Worten: Sie allein bestimmen, wie weit Sie EXBASIC LEVEL II ausbauen wollen, ganz nach Ihren Wünschen. Zwei Standard SOFTMODULE werden zu EXBASIC LEVEL II mitgeliefert, sie enthalten die Befehle SORT/CLEAR u. GOTO X/GOSUB X. In Kürze werden weitere SOFTMODULE lieferbar sein: mehrfachgenaue Arithmetik, erweiterte Graphik, Codieren und Decodieren von Programmen, Matrizenrechnungen, Search & Replace und vieles andere mehr. Schaffen Sie sich schon jetzt die Basis für die Zukunft – mit EXBASIC LEVEL II.
EXBASIC LEVEL II ist der neue Basic-Standard für Commodore Computer. „Extended Level II Basic" stellt für CBM-Computer ein stark erweitertes Level II-Basic – wie es der TRS-80 kennt – zur Verfügung. Mit EXBASIC LEVEL II haben Sie über 75 neue, äußerst leistungsfähige Funktionen für Ihren CBM-Computer, die das Programmieren wesentlich erleichtern und um ein Vielfaches komfortabler gestalten. Die Implementierung von EXBASIC LEVEL II erfolgt in zwei 4kByte Eproms, die einfach in zwei freie Sockel des Computers gesteckt werden und ist völlig unproblematisch. Mit EXBASIC LEVEL II erreichen CBM-Computer eine Leistungsfähigkeit, die bisher als unerreichbar galt. Zudem werden Programme für den TRS-80 mit EXBASIC LEVEL II endlich auch auf CBM lauffähig. Damit steht Ihnen ein riesiges Softwareangebot für Ihren CBM-Computer zur Verfügung. EXBASIC LEVEL II ist bereits weltweit über 5000 Mal verkauft. Wollen Sie es sich leisten, auf die neue Standardisierung zu verzichten?
EXBASIC LEVEL II bekommen Sie bei Ihrem Computerfachhändler
Bundesrepublik Deutschland, Austria, Switzerland: INTERFACE AGE, Kirchenstr. 9, D-8013 Haar bei München, Telefon: 089/465046
USA: DATA DYNAMICS TECHNOLOGY (Div. of INTERFACE AGE), 16704 Marquardt Avenue, Cerritos, CA 90701 (213) 926-9544
England, Scotland, Ireland, Wales: ARFON SOFTWARE Ltd., Cibyn Industrial Estate Caernarfon, Gwynedd, North Wales
Wenn Sie der Meinung sind, daß EXBASIC LEVEL II auch für Sie das Richtige ist – sollten Sie einer anderen Meinung sein, lesen Sie bitte diese Anzeige nochmals – dann besorgen Sie es sich am besten gleich bei Ihrem Computerfachhändler oder benutzen Sie den dreieckigen Bestellcoupon. Bitte geben Sie genau an, für welchen CBM-Computer Sie EXBASIC LEVEL II haben möchten, tragen Sie die entsprechende Ziffer 1 bis 6 bei „Serie" in dem Coupon ein.
1. CBM 2000 (nur mit neuen ROMs!)
2. CBM 3000 mit Graphiktastatur
3. CBM 3000 mit Businesstastatur
4. CBM 4000 mit 9-Zoll-Bildschirm (gleiche Größe wie CBM 2000/3000)
5. CBM 4000 mit 12-Zoll-Bildschirm (gleiche Größe wie CBM 8000)
6. CBM 8000
Der Preis? Er ist – gemessen an den vielfältigen Möglichkeiten, die EXBASIC LEVEL II bietet – so klein, daß wir ihn hier fast vergessen hätten: DM 392,– inkl. MwSt. (unverbindliche Verkaufsempfehlung).
Einsenden an: INTERFACE AGE, Kirchenstr. 9, D-8013 Haar bei München
Ich bestelle hiermit:
☐ Stück EXBASIC LEVEL II Serie ____ (siehe oben)
☐ Mit Anleitungsbuch zum Preis von DM 392,–
☐ Ohne Anleitungsbuch zum Preis von DM 367,–
☐ Stück Anleitungsbücher vorab zum Preis von DM 25,–
Name
Straße/Nr.
PLZ/Ort
Tel.

Dipl.-Phys. Claus M. Müller

Unix, ein Betriebssystem mit Zukunft

Ein kurzer Blick in die Welt der 8-Bit-Rechnersysteme zeigt eine babylonische Sprachverwirrung auf dem Gebiet der Betriebssysteme. Kompatibilität ist nicht einmal zwischen Systemen mit gleichen Prozessoren möglich. Das Argument, eine Standardisierung sei nicht notwendig und behindere die Kreativität, muß in Zukunft ökonomischen Zwängen weichen. Kostenintensive Softwareentwicklung durch professionelle Hersteller lohnt sich nur, wenn der entsprechende Absatz gewährleistet ist. Das ist aber nur möglich, wenn eine weitgehende Standardisierung und Kompatibilität auf der Betriebssystemebene gegeben ist – unabhängig vom verwendeten Prozessor, der ohnehin den Anwender nicht interessiert.

Unter diesem Aspekt ist der aktuelle Trend zu einem einheitlichen Betriebssystem für Rechner mit Prozessoren aus dem 16-Bit-Bereich zu sehen. Abgesehen von einigen Spezialfällen stehen sich zwei Systeme gegenüber: CP/M von Digital Research und Unix von Western Electric (Bell) mit diversen Lizenznehmern.
CP/M ist nach gegenwärtigem Wissensstand des Verfassers nur auf 8086-Systemen implementiert, während sich die kompatible Unix-Familie schon auf allen 16-Bit-Prozessoren sowie auf zahlreichen anderen Rechnern etabliert hat.
Nach Meinung des Verfassers, die durch das derzeitige Marktgeschehen in USA gestützt wird, entscheidet sich die Mehrzahl der Hersteller für Unix, das erfreulicherweise nicht nur ein gutes, sondern auch ein schon lange erprobtes Betriebssystem darstellt.

Die Geschichte von Unix

Die Anfänge von Unix liegen etwa zehn Jahre zurück, als man bei Bell Telephone aus dem Mangel an geeigneten Programmierumgebungen aus der Not eine Tugend machte und ein eigenes System entwickelte. Dieses Projekt bekam eine starke Eigendynamik und wurde im Rahmen einer Forschungsaktivität zum Thema „Portable Betriebssysteme" weiterverfolgt. Um eben diese Portabilität zu erreichen, wurde ein großer Teil des Betriebssystemkerns (ca. 90 %) in der höheren Sprache C implementiert. Der Erfolg gibt dem Konzept recht: vom Z80 bis zur Amdahl reichen die Unix-Gastrechner, wobei der Schwerpunkt naturgemäß auf Rechnern der Minicomputerklasse, wie zum Beispiel PDP 11, liegt.

Verzicht auf spezielle Hardware-Features

Wenn man die Eigenschaften von Unix mit einem Begriff charakterisieren sollte, wäre „neue Einfachheit" sicherlich eine gute Wahl. Diese Einfachheit darf nicht mit Simplizität verwechselt werden, sondern soll vielmehr die Eigenschaft ausdrücken, mit Hilfe klar konzipierter Grundfunktionen flexible Anwendungsmöglichkeiten aufzubauen. „Einfachheit" bezieht sich wesentlich auch auf die Schnittstelle zum Benutzer, der über eine so komfortable und mächtige Kommandosprache verfügt, wie sie in kaum einem anderen Betriebssystem gefunden werden kann.
Durch die Beschränkung auf wichtige Grundfunktionen und den weitgehenden Verzicht auf spezielle Hardwareeigenschaften läßt sich Unix mit wenig Aufwand auf verschiedenen Rechnern implementieren. Dazu trägt die Tatsache bei, daß etwa 90 % des Betriebssystems in der höheren Sprache C geschrieben sind.

Die Hardware-Voraussetzungen für ein Unix-Betriebssystem sind:
□ ca. 80 bis 100 KByte Speicher für den Betriebssystem-Kern (je nach Prozessor),
□ Memory Management Unit,
□ Schneller Plattenspeicher.
Diese Voraussetzungen können (auf Kosten der Effizienz) abgeschwächt werden, wodurch auch Implementierungen auf Z80-Systemen möglich werden.

Komfortable Dateiverwaltung

Kernstück der meisten Betriebssysteme ist die Verwaltung und der Zugriff auf extern gespeicherte Daten, die zu Dateien (Files) zusammengefaßt sind. Man unterscheidet zwischen einer **physikalischen** Ebene, die die Organisation auf dem entsprechenden Speichermedium (Platte, Band etc.) beschreibt und der **logischen** Ebene, die die Struktur der Dateien untereinander angibt.
Unix kennt auf der physikalischen Ebene nur Blöcke von 512 Byte; darauf ist die logische Ebene geschichtet, in der es der Benutzer mit drei Datentypen zu tun hat:
□ **Gewöhnliche Dateien**, in denen Information (Texte, binäre Objekte etc.) gespeichert sind.
□ **Dateiverzeichnisse** (Directories) ermöglichen eine hierarchische Dateistruktur; ein Dateiverzeichnis enthält Verweise auf zugehörige Dateien, die ebenfalls wieder Verzeichnisse sein können. Auf diese Weise können logisch oder organisatorisch zusammengehörige Programme zusammengefaßt werden.
Eine bestimmte Datei wird benannt über Weg-Namen (path-name), d. h. eine Folge von Namen der Dateiverzeichnisse und dem Namen der Datei, zum Beispiel:
/dir1/dira/fname
□ **Spezielle Dateien** (special files).
Jedem Ein/Ausgabegerät ist mindestens eine Spezialdatei zugeordnet, die im Datenverzeichnis /dev zusammengefaßt sind. Um zum Beispiel auf ein Magnetband zu schreiben, wird die Spezialdatei /dev/mt verwendet.
Dieses Konzept bietet mehrere Vorteile: Geräte-I/O und Fileoperation sind im Gebrauch weitgehend äquivalent und Spezialdateien genießen dieselben Schutzmechanismen wie gewöhnliche Dateien.
Die Systemaufrufe für Ein/Ausgabe haben durch die Struktur des Dateisystems einheitliche Schnittstellen für die verschiedenen Geräte und Zugriffsmethoden.

Der Autor ist unabhängiger Software-Consultant mit Sitz in München.

Basisoperationen sind zum Beispiel:

filep = open (name, flag)
  name: Dateiname
  flag gibt die Art des Zugriffs an
  filep dient bei weiteren Systemaufrufen als Dateikennzeichner

Lesen/Schreiben
  n = read (filep, buffer, count)
  n = write (filep, buffer, count)
    count: Anzahl der zu übertragenden Zeichen
    buffer: Adresse eines Speicherbereiches
    n: Anzahl der tatsächlich übertragenen Zeichen

Positionieren
  location = lseek (filep, offset, base)
  positioniert den internen Dateizeiger relativ zum Anfang, Ende oder aktuellen Wert des Dateizeigers.

Mehrere Prozesse gleichzeitig möglich

Unix ist ein System mit der Möglichkeit, mehrere Prozesse gleichzeitig zu verwalten (multitasking). Das Speicherabbild eines Prozesses besteht aus drei Segmenten:
□ Programmcode (schreibgeschützt)
□ Datenbereich
□ Stack-Bereich
Ein Prozeß wird durch den Systemaufruf

processid = fork ()

kreiert, der den gerade gültigen Prozeß aufspaltet.

Interprozeß-Kommunikation

Die Kommunikation zwischen zwei Prozessen wird mittels der schon bekannten Prozeduren read und write durchgeführt. Der Systemaufruf

filep = pipe ( )

erzeugt einen Dateideskriptor filep und eine sog. Pipe. Durch diesen Kanal werden Daten zwischen Prozessen ausgetauscht, die über read und write verarbeitet werden. Kommunizierende Prozesse sehen sich also gegenseitig als Dateien, auf die geschrieben und von welchen gelesen werden kann.
Weitere Systemaufrufe im Zusammenhang mit Prozessen sind:

□ execute (file, arg$_1$ ...arg$_n$)
Der aktuelle Prozeß wird durch das Programm mit Namen „file" und den Parametern arg$_1$ ...arg$_n$ ersetzt.

□ processid = wait (status)
Der aufrufende Prozeß geht solange in den Wartezustand, bis einer der durch ihn kreierten Prozesse seine Ausführung beendet. „processid" ist dann die Kennung dieses Prozesses.

□ exit (status)
beendet den Prozeß, gibt Speicher frei und schließt offene Dateien.

Die Kommandosprache

Die Schnittstelle eines Betriebssystems zum Benutzer ist die Kommandosprache. Sie ist sozusagen die Schale (shell) um den Betriebssystemkern, die das System nach außen darstellt. Der Kommandointerpreter ist selbst ein Prozeß, der Kommandozeilen einliest und ausführt. Eine Kommandozeile hat das Format:

command argument$_1$ ...argument$_n$.

Command ist der Name einer Datei, die ein ausführbares Programm enthält. Der Kommandointerpreter sucht diese Datei und bringt das Programm zum Ablauf.

Wie andere Systeme auch, kennt Unix Standard-Ein/Ausgabe Dateien, üblicherweise Tastatureingabe und Bildschirmausgabe. Diese Standard-Dateien lassen sich über den Kommandointerpreter in einfacher Weise manipulieren: Zum Beispiel ordnet der Aufruf

prog1 > fileout

zur Ausführung des Programms prog1 die Datei „fileout" als Standard-Ausgabe zu.
Analog gilt für Standard-Eingabe

prog > fileinp.

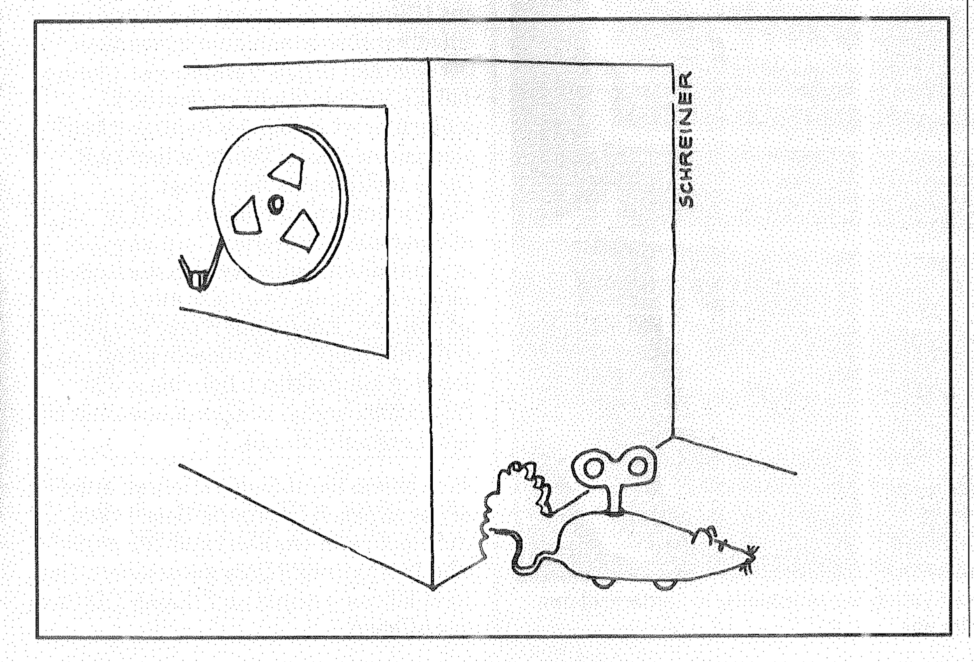
Als „Filter" werden solche Programme bezeichnet, die Daten von der Standard-Eingabe lesen, verarbeiten und auf die Standard-Ausgabe schreiben. Solche Programme lassen sich derart in der Kommandozeile miteinander verketten, daß die Ausgabe des einen Programms zur Eingabe des nächsten Programms wird. Ein Beispiel:

ls | sort | opr

Das Zeichen „ | " dient der Verkettung mehrerer Programme in der oben angegebenen Weise. ‚ls' listet die Dateinamen des aktuellen Dateiverzeichnisses auf, das Programm ‚sort' sortiert diese und ‚opr' druckt sie alle aus. Dieselbe Wirkung hätte man erzielen können mit der Kommandofolge:

```
ls      >temp1
sort    <temp1    >temp2
opr     <temp2    >temp2
```

Weitere Möglichkeiten des Kommandointerpreters sind:
□ Ausführung von Programmen im Background
□ Aufruf von Kommandodateien
□ Passwort-Verwaltung und andere Hilfsfunktionen



Schon der Kommandointerpreter ist nicht mehr Teil des Betriebssystemkerns, sondern ein nichtprivilegierter Prozeß.

Im Lauf der Entwicklung von Unix ist eine ungeheure Zahl von Programmen entwickelt worden, die unter Unix ablaufen. Dazu gehören
□ Compiler für alle wichtigen Sprachen
□ Datenbanksysteme
□ Editoren, Textverarbeitungsprogramme, Aufbereitungsprogramme zur Ausgabe auf Photosatzgeräte
□ Systeme zur Unterstützung großer Programmierprojekte (Programmer's Workbench)
□ Compiler-Compiler (d. h. Compiler-Generatoren) und vieles mehr.

Es würde zu weit führen, alle diese Programme im Detail zu besprechen. Die Unix-Gemeinde ist dermaßen kreativ, daß ein Gesamtüberblick über die verfügbare Software nicht mehr möglich ist.

Unix auf Mikroprozessoren

Mit dem Erscheinen der neuen 16-Bit-Mikros lag es nahe, Unix auch auf solchen Systemen zu implementieren, da diese in der Leistung herkömmlichen Minirechnern vergleichbar sind. Der erste erfolgreiche Versuch wurde natürlich auf einer LSI-11 unternommen, die durch ihre PDP-11-ähnliche Architektur dafür prädestiniert ist. Inzwischen gibt es aber auch Implementierungen für andere Mikros, wobei sich allerdings der jeweilige Name des Betriebssystems verändert hat:
Cromix von Cromemco auf Z80 und M68000 (?)
Idris von Whitesmith auf Z80, M68000, 8086
ONIX von Onyx auf Z8000
XENIX von Microsoft auf 8086, Z8000, M68000
Bis auf Idris sind alle Systeme direkte Implementierungen des Original-Unix; Idris ist eine Unix-kompatible Eigenentwicklung und unterliegt deshalb nicht dem Unix-Lizenzabkommen. Diese Verfügbarkeit auf Prozessoren unterschiedlichen Typs ist eine erfreuliche Entwicklung – bedeutet sie doch, daß im Bereich leistungsfähiger 16-Bit-Systeme die Chance zu einer weitgehenden Standardisierung möglich ist.
Nach den ersten Angeboten der entsprechenden Industrie in USA sieht es auch so aus, als würden Unix und Ableger der Industries Standard. Das hat sicher seinen Grund darin, daß es sehr schwer sein dürfte, dem Markt ein leistungsfähigeres und ausgereifteres Betriebssystem als Unix anzubieten.

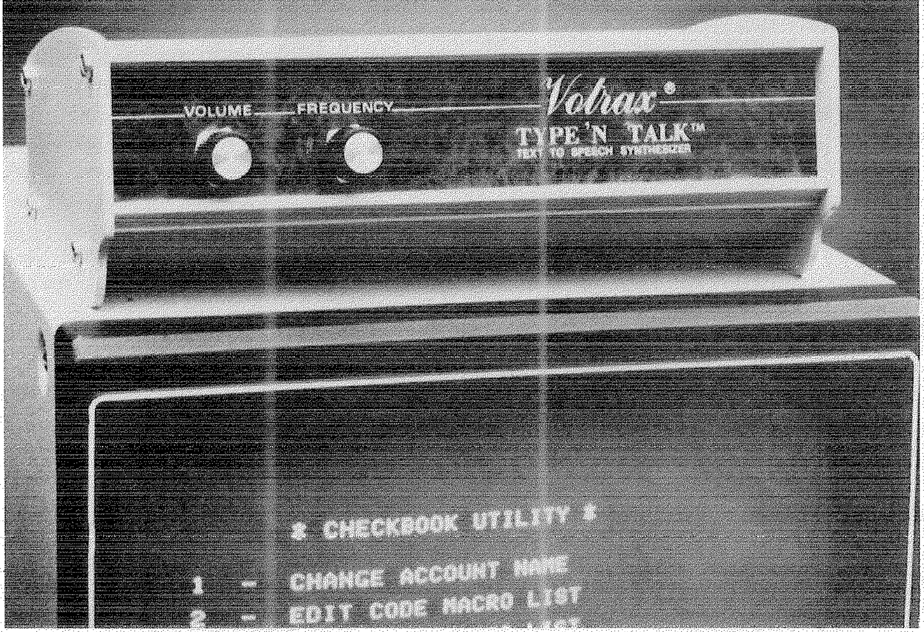
Literatur

[1] Sonderheft über Unix: The Bell Systems Technical Journal. July-August 1978, Vol. 57, No. 6, Part 2.

[2] *Plauger, P. J.*; *Krieger, M. S.*: Unix-like Software Runs on Mini- and Microcomputers. Electronics, 24. März 1981.

Spracherkennung und Sprachsynthese im Vormarsch

Elektronische Spracherkennung und Spracherzeugung werden allmählich auch professionell einsetzbar. Auf mehreren Ausstellungen der letzten Zeit konnte man den Stand der Technik sehen – deutlich wurde dabei, daß zwar die Spracherzeugung schon recht weit fortgeschritten ist, es bei der Spracherkennung aber noch arg hapert.
Während auf der NCC Maryland Computer Services ein „sprechendes Terminal" vorstellte, das die eingetippten oder vom Computer kommenden Zeichen als Lautschrift interpretiert und synthetisch spricht, war bei Lear Siegler Inc. ein Spracherkennungs-Terminal zu sehen, das die Zuordnung beliebiger ASCII-Zeichenfolgen zu in ein Mikrofon gesprochenen Worten ermöglicht, z. B. um einen angeschlossenen Computer mit gesprochenen Befehlsworten zu steuern. Dabei ist ein Vokabular von maximal 100 Worten möglich.



Den ASCII-Code von Computern setzt der Votrax-Synthesizer selbständig in Lautschrift um und gibt ihn als halbwegs verständliche Sprache aus

Mit der erstaunlich niedrigen Bitrate von nur etwa 70...100 Bit/s kommt der Synthesizer „Type 'n Talk" von Votrax aus (*Bild*); auch er setzt die Worte aus einzelnen Lautelementen zusammen. Der Nachteil dieses Verfahrens ist leider, daß die Verständlichkeit deutlich schlechter ist als bei der Synthese komplett gespeicherter Worte, wie sie z.B. Texas Instruments bevorzugt.
Als „Parametric Waveform Coding" bezeichnet die Centrigram Corp. ihr „Lisa"-Verfahren; dabei werden die Sprachparameter für die Synthese alle 50 ms aktualisiert. Komplette Worte werden z. B. auf Floppy-Disk abgespeichert, so daß sich ein praktisch beliebig großer Wortschatz ergibt. Die Sprachqualität ist dabei erstaunlich gut.

Nach einem ähnlichen Verfahren arbeitet das „Speech Computer Board" von Multitech; hier werden rund 400 englische Worte in EPROMs gespeichert. Das Gerät läßt sich direkt an Zilogs MCB-Bus anschließen.

Michael Haßelberg

422 neue Z80-Befehle

Hätten Sie gedacht, daß der Z80 noch vielseitiger ist, als ohnehin schon behauptet wird? Der vorliegende Artikel zeigt diese Möglichkeiten auf. Durch eifriges Experimentieren fand der Autor weitere 422 Opcodes heraus, die den ohnehin schon großen Befehlsvorrat des Prozessors von 694 auf 1116 Opcodes erweitern. Neben neuen Verschiebeoperationen kommen unter anderem weitere Befehle für die indizierte Adressierung hinzu.

Es existieren zunächst acht weitere Shift-Operationen, wobei das betreffende Register (A – L; (HL)) wie beim SLA-Befehl nach links geschoben wird, hierbei wird jedoch von rechts her mit logisch 1 aufgefüllt (*Tabelle 1*).
Die nun folgenden Erläuterungen und Befehle beziehen sich zwar auf das IX-Register, jedoch gilt das Gesagte sinngemäß auch für IY; hierbei ist dann das erste Byte des Opcodes von hex DD in FD zu ändern. Die *Tabellen 2 und 3* geben einen Überblick über zusätzliche Operationen bei den Shift-Befehlen, hierbei wird das Resultat der Operation in eines der Register A...L übertragen. In *Tabelle 2* sind noch einmal die oben beschriebenen Instruktionen bezüglich des Indexregisters IX aufgeführt. Die in *Tabelle 4* aufgezeigten Einzelbitoperationen verhalten sich im Prinzip so wie die Schiebebefehle, denn auch hierbei wird das Resultat der Operation in eines der o. g. Register transferiert. Schließlich geben die *Tabellen 5 und 6* noch einen Überblick über arithmetisch-logische Operationen direkt mit den einzelnen Hälften der Indexregister, wobei mit HX die Bits 8...15 von IX, mit LX die Bits 0...7 von IX gemeint sind. Hierdurch werden diese Indexregister zu vollwertigen Registerpaaren wie HL, DE oder BC.

Diese Liste erhebt natürlich keinen Anspruch auf Vollständigkeit, sie soll nur weitere Anregungen zum Erforschen der inneren Struktur des Z80 geben. Daher ist die Redaktion auch für weitere Ideen und Anregungen jederzeit dankbar.

Tabelle 3: Verschiebeoperationen bezüglich (IX + off) mit anschließendem Transfer in ein Register

	A	L	H	E	D	C	B
RLC (IX + off)	07	05	04	03	02	01	00
RRC (IX + off)	0F	0D	0C	0B	0A	09	08
RL (IX + off)	17	15	14	13	12	11	10
RR (IX + off)	1F	1D	1C	1B	1A	19	18
SLA (IX + off)	27	25	24	23	22	21	20
SRA (IX + off)	2F	2D	2C	2B	2A	29	28
SRL (IX + off)	3F	3D	3C	3B	3A	39	38

Tabelle 1: Verschiebeoperationen SLIA r

	SLIA r
B	CB30
C	CB31
D	CB32
E	CB33
H	CB34
L	CB35
(HL)	CB36
A	CB37

shift left inverted arithmetic r
Cy ◀ 7 ◀ 0 ◀ 1

Tabelle 2: Verschiebeoperationen bezüglich (IX + off) mit anschließendem Transfer in ein Register

XX	B	C	D	E	H	L	(IX+ of)	L
SLIA (IX+of)r	30	31	32	33	34	35	36	37

Opcode: DDcBofXX (of = Offset)

Cy ◀ 7 ◀ 0 ◀ 1
7 * * * 0

Tabelle 4: Einzelbitoperationen bezüglich (LX + off) mit anschließendem Transfer in ein Register

Opcode: DDCBofXX

XX	RES 0, (IX + off)	RES 1, (IX + off)	RES 2, (IX + off)	RES 3, (IX + off)	RES 4, (IX + off)	RES 5, (IX + off)	RES 6, (IX + off)	RES 7, (IX + off)	SET 0, (IX + off)	SET 1, (IX + off)	SET 2, (IX + off)	SET 3, (IX + off)	SET 4, (IX + off)	SET 5, (IX + off)	SET 6, (IX + off)	SET 7, (IX + off)
B	80	88	90	98	A0	A8	B0	B8	C0	C8	D0	D8	E0	E8	F0	F8
C	81	89	91	99	A1	A9	B1	B9	C1	C9	D1	D9	E1	E9	F1	F9
D	82	8A	92	9A	A2	AA	B2	BA	C2	CA	D2	DA	E2	EA	F2	FA
E	83	8B	93	9B	A3	AB	B3	BB	C3	CB	D3	DB	E3	EB	F3	FB
H	84	8C	94	9C	A4	AC	B4	BC	C4	CC	D4	DC	E4	EC	F4	FC
L	85	8D	95	9D	A5	AD	B5	BD	C5	CD	D5	DD	E5	ED	F5	FD
A	87	8F	97	9F	A7	AF	B7	BF	C7	CF	D7	DF	E7	EF	F7	FF

Tabelle 5: Arithmetisch-logische Befehle bezüglich IX

INC HX	DD24	SUB LX	DD95
DEC HX	DD25	SBC HX	DD9C
INC LX	DD2C	SBC LX	DD9D
DEC LX	DD2D	AND HX	DDA4
ADD A, HX	DD84	AND LX	DDA5
ADD A, LX	DD85	XOR HX	DDAC
ADC A, HX	DD8C	XOR LX	DDAD
ADC A, LX	DD8D	CP HX	DDBC
SUB HX	DD94	CP LX	DDBD

Tabelle 6: 8-Bit-Ladebefehle bezüglich IX

LD HX, dd	DD26	LD HX, D	DD62
LD LX, dd	DD2E	LD HX, E	DD63
LD B, HX	DD44	LD HX, LX	DD65
LD B, LX	DD45	LD HX, A	DD67
LD C, HX	DD4C	LD LX, B	DD68
LD C, LX	DD4D	LD LX, C	DD69
LD D, HX	DD54	LD LX, D	DD6A
LD D, LX	DD55	LD LX, E	DD6B
LD E, HX	DD5C	LD LX, HX	DD6C
LD E, LX	DD5D	LD LX, A	DD6F
LD HX, B	DD60	LD A, HX	DD7C
LD HX, C	DD61	LD A, LX	DD7D

Jürgen Plate

„PEEK“ und „POKE“ in Pascal

Viele Benutzer von Pascal glauben, zu Basic oder Assembler zurückkehren zu müssen, wenn sie bestimmte Speicherplätze, etwa für die Bildschirmausgabe oder für Memory-mapped-I/O, gezielt ansprechen wollen. Dabei ist es bei fast allen Pascalsystemen überhaupt kein Problem, gezielt auf den Speicher zuzugreifen. Wie Sie das in Pascal zuwege bringen, sei im folgenden erläutert.

Es handelt sich bei dem geschilderten Verfahren keineswegs um einen unerlaubten Trick, sondern um eine Methode, die vom Pascal-Standard voll gedeckt wird. Um auf den Inhalt einer bestimmten Adresse zugreifen zu können, müßte in Pascal eine Zuweisung eines Wertes an einen Pointer erlaubt sein, denn der Pointer stellt ja nichts anderes dar als eine Adresse.
Normalerweise wird ein Pointer vom Pascalsystem mit einer geeigneten Adresse auf der Halde belegt und eine Wertzuweisung Integer zu Pointer vom Compiler verhindert, was im allgemeinen ja auch sehr sinnvoll ist. Der Kern des Problems ist also, die Typbindung und die Überwachung dieser Typbindung zu überlisten und so einem Pointer eine Adresse zuzuweisen. Diese Möglichkeit wird von Pascal mit den Recordvarianten (variant records) eröffnet. Bei einem vernünftig konstruierten System werden alle Varianten eines Record auf denselben Speicherbereich gelegt und so nur der Platz belegt, den die größte Variante einnimmt. Das können Sie sich zunutze machen, indem Sie eine Integerzahl und einen Pointer „übereinanderlegen“. Nach der Syntax des Pascal User Manual and Report kann das ‚tag field‘ bei der Vereinbarung eines Varianten Records weggelassen werden und es bleibt schließlich die unter Mischtyp unten vereinbarte Datenstruktur übrig.
Bei der Konstruktion der Struktur muß die maschineninterne Darstellung von Integers und Pointern beachtet werden, die gezeigte Darstellung ist für 16-Bit-Adressen und 16-Bit-Integers. Noch einmal, die gezeigte Struktur belegt nur zwei Byte/Speicher, die von Pascal aus einmal als Integerwert und zum anderen als Pointer „gesehen“ werden können. Und hier liegt schon die Lösung des Problems. Durch eine Wertzuweisung an die Komponenten AD des Records kann ich eine beliebige Adresse eintragen und mit der Komponente PT auf den Inhalt der Speicherzelle mit dieser Adresse zugreifen. Unten finden Sie zwei Prozeduren, PEEK zum „Lesen“ des Speichers oder I/O-Ports und POKE zum „Schreiben“ in den Speicher, auf den Bildschirm usw.

Die Prozeduren benötigen zwei globale Typen:

TYPE ADRESSE = 0..65535; oder
ADRESSE = −32767..32767;
BYTE = 0..255;

Wie gesagt, der Kniff der Prozeduren basiert auf einem RECORD mit varianter ‚field list‘.
Diese Recordvereinbarung besagt, salopp gesagt, daß eine Variable dieses Types mal als ganze Zahl (mit eingeschränktem Wertebereich), mal als Zeiger auf ein Byte, also als Adresse betrachtet wird.

```
TYPE MISCHTYP = RECORD CASE BOOLEAN OF
                  TRUE : ( AD : ADRESSE);
                  FALSE: ( PT :  BYTE   )
                END;
```

Mit diesem Wissen sind die Prozeduren ganz einfach zu formulieren:

```
PROCEDURE POKE (A : ADRESSE; B : BYTE);
   TYPE MISCHTYP = RECORD CASE BOOLEAN OF
                     TRUE : ( AD : ADRESSE);
                     FALSE: ( PT :  BYTE   )
                   END;
   VAR X : MISCHTYP;
   BEGIN
   X.AD := A (* ADRESSE BESETZEN *);
   X.PT^:= B (* WERT EINTRAGEN   *)
   END (* POKE *);
```

```
PROCEDURE PEEK (A : ADRESSE; VAR B : BYTE);
   TYPE MISCHTYP = (* siehe oben *);
   VAR X : MISCHTYP;
   BEGIN
   X.AD := A (* ADRESSE BESETZEN *);
   B := X.PT↑(* WERT HOLEN        *)
   END (* PEEK *);
```

Wer PEEK lieber als Funktion hat, muß schreiben:

```
FUNCTION PEEK (A : ADRESSE) : BYTE;
   TYPE MISCHTYP = (* siehe oben *);
   VAR X : MISCHTYP;
   BEGIN
   X.AD := A;
   PEEK := X.PT↑
   END (* PEEK *);
```

Ein paar Einschränkungen sind jedoch zu beachten. Hat der Compiler die Möglichkeit, per Option eine Laufzeitüberwachung ein- oder auszuschalten, dann muß diese unbedingt ausgeschaltet werden (Option T- oder R-). Des weiteren sind die Adressen über 32767 als negative Integerzahlen (Komplement!) einzugeben. Dabei entspricht −32768 der Adresse FFFFH und −1 der Adresse 8000H.
Leider gibt es einige Firmen, die beim Herstellen ihres Compilers nicht an maschinennahe Anwendungen gedacht haben und die Recordvarianten nicht übereinanderlegen. Bei diesen Compilern funktioniert die geschilderte Methode nicht.
Erprobt wurde das Verfahren auf der Pascal-Maschine von Western Digital und auf dem Großrechner CDC Cyber 175.
Ich würde mich freuen, Erfahrungsberichte über die Verwendung des Verfahrens auf Apple, TRS-80, PET und anderen zu erhalten.
Abschließend noch eine nützliche und praktisch erprobte Anwendung. Es wird hier ein Text bildschirmweise kopiert und per Tastendruck weitergeblättert.

```
BEGIN (* HAUPTPROGRAMM *)
ZEILE :=O;
WHILE NOT EOF(INPUT) DO
   BEGIN (* DATEI KOPIEREN *)
   ZEILE := ZEILE + 1;
   COPYLINE(INPUT,OUTPUT);
   IF ZEILE MOD 20 = O THEN
      (* WARTE BIS EINE TASTE GEDRUECKT WURDE *)
      REPEAT
         PEEK(TASTENPUFFER,DUMMY)
      UNTIL DUMMY   O;
   END;
END.
```

APPEND für TRS-80 L II

Um ein Programmteil an ein im Speicher vorhandenes Programm anzufügen, genügt es, das folgende Basicprogramm hinter das vorhandene Programm einzugeben und zu starten. Es belegt den Inhalt des Programmanfangszeigers in hex 40A4,5 mit der letzten Speicheradresse des aktuellen Programms. Diese läßt sich aus dem Inhalt des Variablenspeicherzeigers 40F9, A H berechnen, indem davon 2 subtrahiert wird.

```
A = PEEK (16633) : B = PEEK
(16634) : A = A-2
IF A < 0 THEN A = A + 256 : B = B-1
POKE 16548, A : POKE 16549, B : END.
```

Betrachtet man sich nun seinen Programmspeicher z. B. mit dem LIST-Befehl, so scheint er leer zu sein. Er ist jedoch nur verdeckt und ein Unterprogramm kann mit CLOAD zugeladen werden. Zu beachten ist dabei nur, daß die Zeilennummern des zugeladenen Programms größer sind, als die des verdeckten Programms. Dies ist bereits bei der Erstellung von Unterprogrammen dadurch zu berücksichtigen, daß diese mit genügend großen Zeilennummern beginnen.
Um das verdeckte Programm wieder zu aktivieren, muß in den Programmanfangszeiger wieder die Adresse 42E9 H eingegeben werden. Dies bewerkstelligt die Befehlsfolge:

```
POKE 16548,233 : POKE 16549,66.
```

Abschließend sind noch die drei Basiczeilen des Zuladeprogramms zu löschen, sofern sie nicht im Befehlsmodus eingegeben wurden. Werner Linsler

Berichtigung: Das Innenleben des MZ-80K

mc 1981, Heft 4, Seite 39

Die Bildunterschrift muß richtig lauten: „...B bezeichnet ein Byte, wobei jedes Bit als S für Null und L für Eins gesendet wird, das höchstwertige unüblicherweise zuerst. Jedem Byte ist ein L als Startbit vorangestellt." Im Bild ist die Anfangsimpulsfolge korrekterweise so: 22000 × S, 40 × L, 40 × S beim Deskriptor, 11000 × S, 20 × L, 20 × S beim Datensegment.

Hans Kohorst

Der geknackte CBM

Während z. B. Tandy klammheimlich ab und zu die ROM-Software seiner Computer ändert und auch selbst keinerlei Dokumentation darüber verbreitet, existieren bei Commodores CBM heute drei Betriebssysteme (2000/3000/4000), die wenigstens zum Teil in den Handbüchern dokumentiert sind. Will man etwas tiefer einsteigen oder Maschinenprogramme adaptieren, reichen die Handbuch-Angaben aber meist nicht aus. Hier folgt deshalb eine Gegenüberstellung von ROM-Adressen in den Versionen 3000 und 4000 des CBM; letztere ist mit dem CBM 8000 identisch.

Basic 3.0	Basic 4.0	Beschreibung
C000-C045	B000-B065	Tabellen der Basic-Befehle
C046-C073	B066-B093	Tabellen der Basic-Funktionen
C074-C091	B094-B0B1	Hierarchie und „Action-Adressen"
C092-C192	B0B2-B20C	Tabelle der Basic-Worte
C193-C2A9	B20D-B321	Error-Meldungen
C2AA-C2D7	B322-B34F	Sucht im Stack nach FOR- oder GOSUB-Aktivitäten
C2D8-C31A	B350-B392	Schafft Raum für neue Zeile
C31B-C327	B393-B39F	Test: Stack zu tief
C328-C354	B3A0-B3CC	Verfügbaren Speicher prüfen
C355	B3CD	Holt Fehlermeldung (ab C193) und vollzieht
C389-C3AA	B3FF-B41E	Warmstart: „READY."
C3AB-C441	B41F-B4B5	Bearbeitet neue Basic-Zeile
C442-C46E	B4B6-B4E1	Verketten von Basic-Zeilen nach Löschen bzw. Einfügen
C46F-C494	B4E2-B4FA	Holt Zeile von Tastatur
C495-C52B	B4FB-B5A2	Keywords vergleichen und durch 1-Byte-Befehl ersetzen
C52C-C55A	B5A3-B5D1	Sucht nach vorhandenen Zeilennummern
C55B	B5D2	Führt NEW aus und
C577-C5A6	B5EC-B621	generiert CLR
C5A7-C5BA	B622-B62F	Startet Basic-Programm
C5B5-C657	B630-B6DD	Generiert LIST
C658-C6FF	B6DE-B784	Generiert FOR
C700-C72F	B785-B7B6	Ausführung eines Basic-Statements
C730-C73E	B7B7-B7C5	Generiert RESTORE
C73F-C76A	B7C6-B7ED	Generiert STOP oder END
C76B-C784	B7EE-B807	Generiert CONT
C785-C78F	B808-B812	Generiert RUN
C790-C7AC	B813-B839	Generiert GOSUB
C7AD-C7D9	B83A-B85C	Generiert GOTO
C7DA	B85D	Generiert RETURN: dann
C7F3-C80D	B883-B890	Generiert DATA: Überspringen der Zeile
C80E-C810	B891-B893	Sucht nach nächstem Statement
C811-C82F	B894-B8B2	Sucht nach nächster Zeile
C830	B8B3	Generiert IF: wenn, dann
C843-C852	B8C6-B8D5	Generiert REM: überspringen
C853-C872	B8D6-B8F5	Generiert ON
C873-C8AC	B8F6-B92F	Holt Integer-Zahl
C8AD-C927	B930-BA87	Generiert LET
C98B-C990	BA88-BA8D	Generiert PRINT#
C991-C9A4	BA8E-BAA1	Generiert CMD
C9A5-CA1B	BAA2-BB1C	Generiert PRINT
CA1C-CA38	BB1D-BB39	Ausgabe eines STRINGS (Adresse im Akku und Y-Reg.)
CA39-CA4E	BB3A-BB4B	Ausgabe eines Zeichens (Inhalt des Akkus)
CA4F-CA7C	BB4C-BB79	Bearbeitung falscher Eingaben
CA7D-CAA6	BB7A-BBA3	Generiert GET
CAA7-CAC0	BBA4-BBBD	Generiert INPUT#
CAC1-CAF9	BBBE-BBF4	Generiert INPUT
CAFA-CB06	BBF5-BC01	Eingabe annehmen und beantworten
CB07-CBFB	BC02-BCF6	Generiert READ
CBFC-CC1F	BCF7-BD18	Meldung EXTRA IGNORED/REDO FROM START
CC20-CC78	BD19-BD71	Generiert NEXT
CC79-CC9E	BD72-BD97	Ausgabe: TYPE MISMATCH (wenn erforderlich)
CC9F-CDEB	BD98-BEE8	Holt Strings und numerische Ausdrücke
CDEC-CDF1	BEE9-BEEE	Behandelt Ausdruck in Klammern
CDF2-CE02	BEEF-BEFF	Sucht nach Komma
CE03-CE07	BF00-BF0B	Ausgabe SYNTAX ERROR: verläßt Programm
CE08-CE0E	C047-C085	Erkennt Funktion und stellt Bezug dazu her
CE0F-CE88	BF8C-C046	Sucht nach Variablen-Namen
CEC8-CEF7	C086-C0B5	Generiert OR und AND
CEF8-CF5F	C0B6-C11D	Führt Vergleiche aus
CF60-CF6C	C11E-C12A	Generiert DIM
CF6D-CFF6	C12B-C1BF	Sucht nach Variablen
D001-D077	C1C0-C2C7	Setzt neue Variable
D078-D088	C2C8-C2D8	Subroutine für Array-Pointer
D089-D08C	C2D9-C2DC	Zahl 32768 in binärer Form
D08D-D0AB	C2DD-C2FB	Umrechnung Fließkomma/Integer
D0AC-D227	C2FC-C447	Suche/Aufbau eines Arrays
D228-D258	C477-C4A7	Bearbeitet ARRAY
D259	C4A8	Generiert FRE
D26D-D279	C4BC-C4C8	Wandelt Integer in Fließkomma um
D27A-D27F	C4C9-C4CE	Generiert POS
D280-D28C	C4CF-C4DB	Prüft, ob direkt. oder indirekt. Befehl. Ausg.: ILLEGAL DIRECT
D28D-D2BA	C4DC-C509	Generiert DEF
D2BB-D2CD	C50A-C51C	Überprüft FN-Syntax
D2CE-D33E	C51D-C58D	Bearbeitet FN
D33F-D34E	C58E-C59D	Generiert STR$
D34F-D360	C59E-C5AF	Setzt STRING-Vektor
D361-D3CD	C5B0-C61C	Sucht nach STRING-Elementen
D3EC-D3FF	C61D-C669	Schafft Freiraum für String
D400-D496	C66A-C74E	Entfernt überflüssige Strings
D517-D553	C74F-C78B	Concatenation-Routine (String-Addition)
D554-D57C	C78C-C7B4	Speichert String
D57D-D5B4	C7B5-C810	Überspringt nicht gebrauchte Strings
D5B5-D5C5	C811-C821	Säubert Descriptor-Stack
D5C6-D5D9	C822-C835	Generiert CHR$
D5DA-D605	C836-C861	Generiert LEFT$

Basic 3.0	Basic 4.0	Beschreibung
D606-D610	C862-C86C	Generiert RIGHT$
D611-D63A	C86D-C896	Generiert MID$
D63B-D655	C897-C8B1	Holt String-Data
D656-D65B	C8B2-C8B7	Generiert LEN
D65C-D664	C8B8-C8C0	Holt Länge des Strings
D665-D674	C8C1-C8D0	Generiert ASC
D675-D686	C8D1-C8E2	Holt 1-Byte-String aus Basic
D687-D6C5	C8E3-C920	Generiert VAL
D6C6-D6D1	C921-C92C	Holt 2 Parameter für POKE und WAIT
D6D2-D6E7	C92D-C942	Prüft, ob Bereich 0-65535 für POKE oder WAIT eingehalten
D6E8-D706	C943-C959	Generiert PEEK
D707-D70F	C95A-C962	Generiert POKE
0710-D72B	C963-C97E	Generiert WAIT
D72C-D732	C97F-C985	Addiert 0.5 zu Akku 1
D733-D744	C986-C997	Generiert Subtraktion
D76E-D852	C998-CA7C	Generiert Addition
D853-D889	CA7D-CAB3	Komplementiert Akku 1
D88A-D88E	CAB4-CAB8	Overflow
D88F-D8C7	CAB9-CAF1	Multipliziert ein Byte
D8C8-D8F5	CAF2-CB1F	Enthält Fließkomma-Konstante
D8F6-D936	CB20-CB5D	Generiert LOG
D937-D964	CB5E-CBC1	Generiert Multiplikation
D998-D9C2	CBC2-CBEC	Lädt Akku 2 mit Memory (Adresse in Akku und Y-Register)
D9C3-D9DF	CBED-CC09	Testet und justiert Akku 1 und 2
D9E0-D9ED	CC0A-CC17	Behandelt Overflow und Underflow
D9EE-DA04	CC18-CC2E	Multiplikation mit 10
DA05-DA09	CC2F-CC33	10 in binärer Form
DA0A-DA12	CC34-CC3C	Division durch 10
DA1E-DAAD	CC3D-CC44	Generiert Division durch x
DA13-DA1D	CC45-CCD7	Generiert Division in x und y
DAAE-DAD2	CCD8-CCFC	Lädt Akku 1 aus Memory
DAD3-DB07	CCFD-CD31	Lädt Memory mit Akku 1
DB08-DB17	CD32-CD41	Bringt Akku 2 nach Akku 1
DB18-DB26	CD42-CD50	Bringt Akku 1 nach Akku 2
DB27-DB36	CD51-CD60	Rundet Akku 1
DB37-DB44	CD61-CD6E	Holt Vorzeichen von Akku 1
DB45-DB63	CD6F-CD8D	Generiert SGN
DB64-DB66	CD8E-CD90	Generiert ABS
DB67-DBA6	CD91-CDD0	Vergleicht Akku 1 mit Memory
DBA7-DBD7	CDD1-CE01	Wandelt Fließ- in Fest-Komma
DBD8-DBFE	CE02-CE28	Generiert INT
DBFF-DC89	CE29-CEB3	Wandelt ASCII-String in Fließkomma-Zahl
DC8A-DCBE	CEB4-CEE8	Holt neue ASCII-Stelle
DCBF-DCCD	CEE9-CEF8	Enthält Konstante
DCCE	CF78	Ausdruck IN, dann
DCD9-DCE8	CF7F-CF92	Ausdruck Basic-Zeilennummer
DCE9-DE1C	CF93-D0C6	Wandelt Fließkommazahl in ASCII
DE1D-DE5D	D0C7-D107	Enthält Konstanten
DE5E-DE67	D108-D111	Generiert SQR
DE68-DEA0	D112-D14A	Generiert Exponential-Funktionen
DEA1-DEAB	D14B-D155	Generiert Negation
DEAC-DED9	D156-D183	Enthält Konstanten
DEDA-DF2C	D184-D1D6	Generiert EXP
DF2D-DF76	D1D7-D220	Testet Mehrfach-Funktionen
DF77-DF7E	D221-D228	Enthält RND-Konstanten
DF7F-DFD7	D229-D281	Generiert RND
DFD8-DFDE	D282-D288	Generiert COS
DFDF-E027	D289-D2D1	Generiert SIN
E028-E053	D2D2-D2FD	Generiert TAN
E054-E08D	D2FE-D32B	Enthält Konstante
E08E-E0BB	D32C-D35B	Generiert ATN
E0BC-E0F8	D35C-D398	Enthält Konstante
E0F9-E110	D399-D3B5	CHRGET für Zero-Page
E116-E1B6	D3B6-D471	Basic-Kaltstart

Basic 3.0	Basic 4.0	Beschreibung
E1B7-E1DD	D448-D471	Enthält BYTES FREE und ###COMMODORE BASIC###
Diese	D7AC-D802	Generiert RECORD
Funktionen	D803-D837	Testet Disk-Parameter
sind im	D838-D872	Dummy Disk Control Message
Basic 3.0	D873-D919	Generiert CATALOG oder DIRECTORY
nicht	D91A-D92E	Ausgabe
enthalten.	D92F-D941	Sucht freie Sekundäradresse
	D942-D976	Generiert DOPEN
	D977-D990	Generiert APPEND
	D991-D9D1	Holt Disk Status
	D9D2-DA06	Generiert HEADER
	DA07-DA30	Generiert DCLOSE
	DA31-DA64	Führt Disk-Aufzeichnung durch
	DA65-DA7D	Generiert COLLECT
	DA7E-DAA6	Generiert BACKUP
	DAA7-DAC6	Generiert COPY
	DAC7-DAD3	Generiert CONTACT
	DAD4-DB0C	Fügt Kommando ein
	DB0D-DB39	Generiert DSAVE
	DB3A-DB65	Generiert DLOAD
	DB66-DB98	Generiert SCRATCH
	DB99-DB9D	Tested DIRECT COMMAND
	DB9E-DBD6	Ausgabe: ARE YOU SURE?
	DBD7-DBE0	Ausgabe: BAD DISK
	DBE1-DBF9	Löscht DS$ und ST
	DBFA-DC67	Setzt Disk-Kommando um
	DC68-DE29	Führt Basic-DOS aus
	DE2C-DE48	Holt DEVICE-Nummer
	DE49-DE86	Holt File-Namen
	DE87-DE9C	Holt Variablen-Parameter
* Nur Einstiegs-Punkte für den Bereich E1DE (E000) bis E6E4 (E600) *		
E1DE	E000	Initialisiert Register (Clear screen, Reset-Routine)
E285	E0A7	Nimmt Eingabe von Tastatur an
E2F2	E116	Nimmt Eingabe vom Bildschirm an
E3D8	E202	Gibt ein Zeichen auf Bildschirm
E61B	E442	INTERRUPT (Einstieg)
E62E	E455	Hardware-Interrupt-Routinen (Uhr, Cursor, Tastenfeld)
E6E4	E600	INTERRUPT (Ausgang)
E76A-E7FF	D717-D7AB	MLM subroutine
F000-F0B5	F000-F0D1	File-Messages
F0B6-F0B9	F0D2-F0D4	Sendet „TALK"
F0BA-F0BB	F0D5-F0D6	Sendet „LISTEN"
F0BC-F0ED	F0D7-F108	IEEE-Steuerzeichen
F0EE-F127	F109-F142	Sendet 1 Byte zum IEEE-Bus
F128-F135	F143-F150	Sendet Byte, löscht ATN
	F151-F16B	Option: TIMEOUT oder WAIT
F136-F13F	F16C-F16F	Ausgabe: DEVICE NOT PRESENT
F140-F155	F170-F184	Lesetakt, löscht Steuerleitungen
F156-F163	F185-F192	Sendet File-Messages
F164-F16E	F193-F19D	Sendet Byte, löscht Steuerleitung
F16F-F17E	F19E-F1AD	Sendet IEEE-Zeichen
F17F-F18B	F1AE-F1BF	Schaltet IEEE-Device ab
F18C-F1D0	F1C0-F204	Holt Byte vom IEEE-Bus (INPUT)
F1D1-F1E0	F205-F214	Holt ein Byte (GET)
F1E1-F231	F215-F265	Holt ein Byte (INPUT)
F232-F26D	F266-F2A1	Gibt ein Byte aus
F26E-F283	F2A2-F2B5	Verläßt alle Files (kein CLOSE)
F284-F28C	F2B6-F2C0	Restore benutzte I/O Devices
F28D-F2A8	F2C1-F2DC	Sucht nach File-Daten
F2A9-F300	F2DD-F334	Generiert CLOSE
F301-F30E	F335-F342	Abfrage: RUN/STOP-Taste
F30F-F314	F343-F348	Bearbeitet RUN/STOP-Taste
F315-F31C	F349-F350	Prüft Direktmodus und gibt Meldung aus

Basic 3.0	Basic 4.0	Beschreibung
F31D-F321	F351-F355	Test, ob Direktmodus
F322-F3C1	F356-F400	Programmlade-Unterprogramm
F3C2-F409	F401-F448	Generiert LOAD
F40A-F42D	F449-F46C	Ausgabe: SEARCHING
F42E-F43D	F46D-F47C	Ausgabe: LOADING oder VERIFYING
F43E-F45F	F47D-F4A4	Holt LOAD/SAVE-Parameter
F466-F493	F4A5-F4D2	Sendet NAME zum IEEE-Bus
F494-F4B6	F4D3-F4F5	Findet speziellen Tape-Header
F4B7-F4CD	F4F6-F50C	Generiert VERIFY
F4CE-F50D	F50D-F55F	Holt OPEN/CLOSE-Parameter
F521-F5A5	F560-F5E4	Generiert OPEN
F5A6-F5D9	F5E5-F618	Findet nächsten Tape-Header
F5DA-F63B	F619-F67A	Schreibt Tape-Header
F63C-F655	F67B-F694	Holt Start/End-Adr. des Headers
F656-F66B	F695-F6AA	Setzt Buffer-Adresse
F66C-F683	F6AB-F6C2	Setzt Buffer-Start- und End-Adr.
F684-F68C	F6C3-F6CB	Generiert SYS
F68D-F69D	F6CC-F6DC	Setzt Tape-Write (Start und Endadr.)
F69E-F728	F6DD-F767	Generiert SAVE
F729-F76F	F768-F7AE	Stellt Uhr nach
F770-F7BB	F7AF-F7FD	Ordnet INPUT-Device zu
F7BC-F805	F7FE-F84A	Ordnet OUTPUT-Device zu
F806-F811	F84B-F856	Tape-Buffer-Pointer betätigen
F812-F834	F857-F879	Wartet, bis PLAY gedrückt
F835-F846	F87A-F88B	Testet, ob Recorder angeschaltet
F847-F854	F88C-F899	Wartet, bis RECORD und PLAY gedrückt
F855-F885	F89A-F8CA	Initial. Tape-Read (liest 192 B)
F886-F89A	F8CB-F8DF	Initial. Tape-Write (schr. 192 B)
F89B-F8E5	F8E0-F92A	Tape-I/O-Routine
F8E6-F8EF	F92B-F934	Wartet auf norm. Interrupt (wenn I/O beendet)
F8F0-F8FF	F935-F944	Prüft, ob STOP-Taste gedrückt
F900-F930	F945-F975	Read-Timing-Subroutine
F931-FA56	F976-FA9B	Liest Bits zum Tape
FA57-FB75	FA9C-FBBA	Liest Zeichen vom Tape
FB76-FB7E	FBBB-FBC3	Löscht Tape-Read-Adresse
FB7F-FB83	FBC4-FBC8	Error-Flag nach ST
FB84-FB92	FBC9-FBD7	Löscht Zähler für neues Byte
FB93-FBAE	FBD8-FBF3	Schreibt ein Bit auf Band
FBAF-FC40	FBF4-FC85	Tape-Write-Routine
FC41-FC7A	FC86-FCBF	Schreibt Tape-Leader
FC7B-FC95	FCC0-FCDA	Ende der Tape-Aufz., Interrupt-Vektor rückspeichern
FC96-FCA5	FCDB-FCEA	Setzt Interrupt-Vektor
FCA6-FCB3	FCEB-FCF8	Schaltet Recorder-Motor aus
FCB4-FCC5	FCF9-FD0A	Überprüft die Prüfsumme
FCC6-FCD0	FD0B-FD15	Erhöht LOAD/SAVE-Pointer
FCD1-FCFD	FD16-FD4B	Power-On-Reset
FD01-FD10	FD4C-FD5C	Tabelle der Interrupt-Vektoren
FD11-FFB0	D472-D716	Maschinensprache-Monitor (TIM)

Sprungtabelle:

Basic 3.0	Basic 4.0	Beschreibung
	FF93-FF9E	CONTACT, DOPEN, DCLOSE, RECORD
	FF9F-FFAA	HEADER, COLLECT, BACKUP, COPY
	FFAB-FFB6	APPEND, DSAVE, DLOAD, CATALOG
	FFB7-FFBC	RENAME, SCRATCH
	FFBD	Holt Disk-Status
FFC0	FFC0	OPEN
FFC3	FFC3	CLOSE
FFC6	FFC6	Setzt INPUT-Device
FFC9	FFC9	Setzt OUTPUT-Device
FFCC	FFCC	Restored benutzte I/O-Devices
FFCF	FFCF	INPUT ein Byte
FFD2	FFD2	OUTPUT ein Byte
FFD5	FFD5	LOAD
FFD8	FFD8	SAVE
FFDB	FFDB	VERIFY
FFDE	FFDE	SYS
FFE1	FFE1	Testet Stop-Taste
FFE4	FFE4	Holt ein Byte
FFE7	FFE7	Verläßt alle Files (kein CLOSE)
FFEA	FFEA	Stellt Uhr nach
FFFA-FFFF	FFFA-FFFF	ROM-Vektoren: NMI, RESET, INT

Adressen, die unter der Rubrik Basic 3.0 nicht erscheinen, sind nur für Disketten-Operationen bestimmt.
Bei Verwendung von Betriebs-Subroutinen ist darauf zu achten, daß die erste angegebene Adresse nicht immer mit der Einstiegsadresse übereinstimmen muß.

Beispiel:

CA39 (BB3A) Ausgabe eines Zeichens
Um diese Routine zu benutzen, ist folgendes notwendig:
1. Laden des Akkus (direkt oder indirekt) mit dem auszugehenden Wert.
2. Ansprung der Routine nach Adresse CA45 (BB46)

Aber:

CA1C (BB1D) Ausgabe eines Strings
Hier ist folgendes erforderlich:
1. Akku mit Lower-Order-Byte der Adresse des Strings laden.
2. Y-Register mit Higher-Order-Byte der Adresse des Strings laden.
3. Aufruf der Routine durch CA1C (BB1D).
(Der String *muß* mit 00 enden!)

Beispiel-Programme:

1. Ausgabe von 256 Byte

```
033A   A2 00      LDX # $ 00       Schleifenzähler
033C   BD 00 10   LDA $ 1000,X     Byte in Akku laden
033F   20 45 CA   JSR $ CA45       Ausgabe eines Bytes
0342   E8         INX              Schleifenzähler + 1
0343   D0 F7      BNE $ 033C       Schleifenzähler = 0?
0345   60         RTS              Zurück nach Basic
```

Die Bytes der Adressen 1000 bis 10FF werden auf den Bildschirm ausgegeben.

2. Ausgabe eines Strings

```
033A   A9 00      LDA #$ 00        Akku mit L-Byte laden
033C   A0 10      LDY #$ 10        Y-Register mit H-Byte
033E   20 1C CA   JSR $ CA1C       Ausgabe d. adress. Str.
0341   60         RTS              Zurück nach Basic

1000   0D 43 42 4D 20 42 41 53
1009   49 43 00 AA AA AA AA AA
```

Die Bytes der Adressen 1000 bis 100A werden als String ausgegeben.
(0D = Line-Feed, AA = Inhalt nach dem Einschalten; nicht wichtig)

Johannes Leckebusch

Was ist UCSD-Pascal?

UCSD-Pascal ist die Pascal-Implementierung der *U*niversity of *C*alifornia, *S*an *D*iego. Eigentlich handelt es sich um ein ganzes Betriebssystem (UCSD-P-System), das inzwischen in der Version IV.0 die Sprachen Pascal, Fortran-77 und Basic (Compiler) sowie Macro-Assembler bzw. Cross-Assembler für 8080, Z-80, 6502, 6800, 6809, 9900, Z8 und PDP-11/LSI-11 umfaßt.

Man erhält UCSD-Pascal entweder als adaptierbares System (dann muß ein Installations-Dialog unter CP/M ausgeführt werden, der in der sehr umfangreichen Dokumentation ausführlich beschrieben ist) oder fertig konfiguriert für einen bestimmten Rechner. Einmal installiert, bildet es ein geschlossenes System (im Gegensatz zu anderen Implementierungen von Pascal, die beispielsweise unter CP/M laufen). Das wird oft als Nachteil empfunden, weil es nicht so einfach ist, bereits unter CP/M vorhandene Programme zu benutzen. Allerdings gibt es Hilfsprogramme, um Source zwischen CP/M- und UCSD-Systemen hin- und hertransportieren zu können. Das Übersetzen und Binden im UCSD-System ist dann sehr viel einfacher als in CP/M, vor allem funktioniert es wirklich! Der Autor hat beobachtet, daß mit CP/M vertraute Anwender höchst überrascht waren, wie komfortabel und benutzerfreundlich das UCSD-Betriebssystem ist.

Pascal und Maschinensprache

Die Bezeichnung „P-System" besagt, daß das UCSD-Pascal einen primitiven Interpreter für den Maschinencode einer idealisierten CPU – der P-Maschine – enthält. Dieser Maschinencode wurde sorgfältig optimiert, so daß der unvermeidliche Geschwindigkeitsverlust gegenüber direkt compilierten Sprachen überaschend gering ist. Dieser kleine Nachteil verkehrt sich schließlich sogar in einen Vorteil, wenn man eine Hardware-P-Maschine benutzt, d. h. einen Prozessor, der den P-Code als seine eigene Maschinensprache verarbeitet (Western-Digital-Microengine), dann wird auch noch der sonst vom Interpreter benutzte Speicherplatz verfügbar! In [1] findet man einen „Benchmark"-Test für höhere Programmiersprachen auf der Basis eines Primzahlprogramms nach dem „Sieb des Eratosthenes". Dort schneidet UCSD als bestes P-Code-Pascal ab, die Compilierungszeiten sind weit günstiger als die von Implementierungen, die Zielmaschinencode erzeugen (und zusätzlich einen Assembler- und Linker-Lauf benötigen).

UCSD wird von vielen in der Maschinensprache des jeweiligen Prozessors geschriebenen Funktionen unterstützt, etwa zum schnellen Füllen oder Kopieren von Arrays, so daß es bei deren geschickter Ausnutzung in manchen Fällen kaum einem in Assembler geschriebenen Programm nachsteht (z. B. Texteditor). Auf der anderen Seite ist der P-Code kompakter als gewöhnlicher Maschinencode, weil er sprachbezogen gestaltet ist, und, das ist der größte Vorteil, **compilierte** Programme können zwischen Systemen mit unterschiedlicher CPU übertragen werden (entweder per Diskette oder einfach über RS-232)! Natürlich nur dann, wenn die zugrunde liegenden P-Maschinen übereinstimmen. Der Compiler selbst arbeitet mehrfach schneller als der Lauf eines Compilers, Assemblers und Linkers beispielsweise in Pascal/Z oder Pascal/MT+ (siehe [1]).

Schließlich können mit dem verfügbaren Makroassembler externe Prozeduren in Assembler geschrieben werden. Ihre Argumente empfangen sie über PUSH-Befehle vom Maschinenstack (Z-80) und legen die Resultate dort mit POP wieder ab. Der Anwender ist vollkommen von Überlegungen hinsichtlich Speicher- und Startadressen befreit, er kann lokale und globale Datenobjekte im Assembler deklarieren und die Assemblerroutinen in Pascal wie Pascal-Prozeduren aufrufen bzw. wie Pascal-Funktionen benutzen. Der Rücksprung aus dem Assembler-Programm erfolgt durch einen einfachen RETURN- oder JP-Befehl zu der vorher vom Stack geretteten Adresse. Das Einfügen in den P-Maschinen-Code besorgt ein Linker quasi automatisch. In der Source wird lediglich eine Vereinbarung der Form „procedure assproc (arg1, arg2: integer); external;" oder ähnlich eingefügt.

Die Benutzeroberfläche des Systems

Das Betriebssystem ist ungewöhnlich komfortabel und betriebssicher. Es arbeitet weitgehend mit Eintasten-Befehlen. Beispielsweise erspart einem das Konzept des „Arbeitsfiles" (Workfile) jegliches Merken von Aufrufformaten von Assembler, Linker usw. und das Merken von File-Namen. Außerdem startet sich der Editor wesentlich schneller mit dem Arbeitstext als etwa der WordStar unter CP/M. Der UCSD-Screen-Editor ist für Programmeditierungen sogar noch komfortabler als der genannte Editor. Man kann kaum etwas zerstören; die Betriebssystem-Utilities fragen nach fehlenden Informationen oder geben eine Erklärung ab, wenn sie den gewünschten Auftrag nicht erledigen können. Die Disketten können über ihren Namen oder, wahlweise, über das Laufwerk, in dem sie sich befinden, angesprochen werden. Alle Files in der Directory tragen ein Tagesdatum (das man tunlichst beim Systemstart im Filer, s. u. eingibt). Eine unschätzbare Dokumentationshilfe! Ins Schleudern kommt das System nur dann, wenn man in zwei verschiedenen Laufwerken Disketten mit dem gleichen Namen hat – aber es stößt dann auch laufend Proteste aus und weist auf diesen Umstand hin! Zugegeben: man muß erst einmal die Dokumentation lesen. Eine ausgezeichnete Hilfestellung bietet [3].

Das System umfaßt Compiler (Kommando: „C"), Screen-Editor („E"), Filer (Filer-Manager („F"), zum Kopieren von Files, Disketten, Umbenennen von Files, Ausdrucken von Textfiles und ...), Linker, Library-Manager (Aufbau und Verwaltung von Programm-Bibliotheken), Assembler („A"), Disk-Formatierer und anderes mehr. Der Amateur kann das System schon mit 48 KByte RAM „fahren", für professionelle Anwendungen sind aber 56 oder besser 64 KByte und zwei Laufwerke erforderlich.

Bekannt sind dem Autor bislang fertige Implementierungen auf den Personal-Computern von Tandy (Model I und Model II), Apple, Exidy (Sorcerer), Ohio (Challenger), Intertec (Superbrain, von Datalex-Company, San Francisco i. Version IV.0 – siehe Byte, July 81, S. 299), aber das sind natürlich nicht alle (konsultieren Sie amerikanische Zeitschrif-

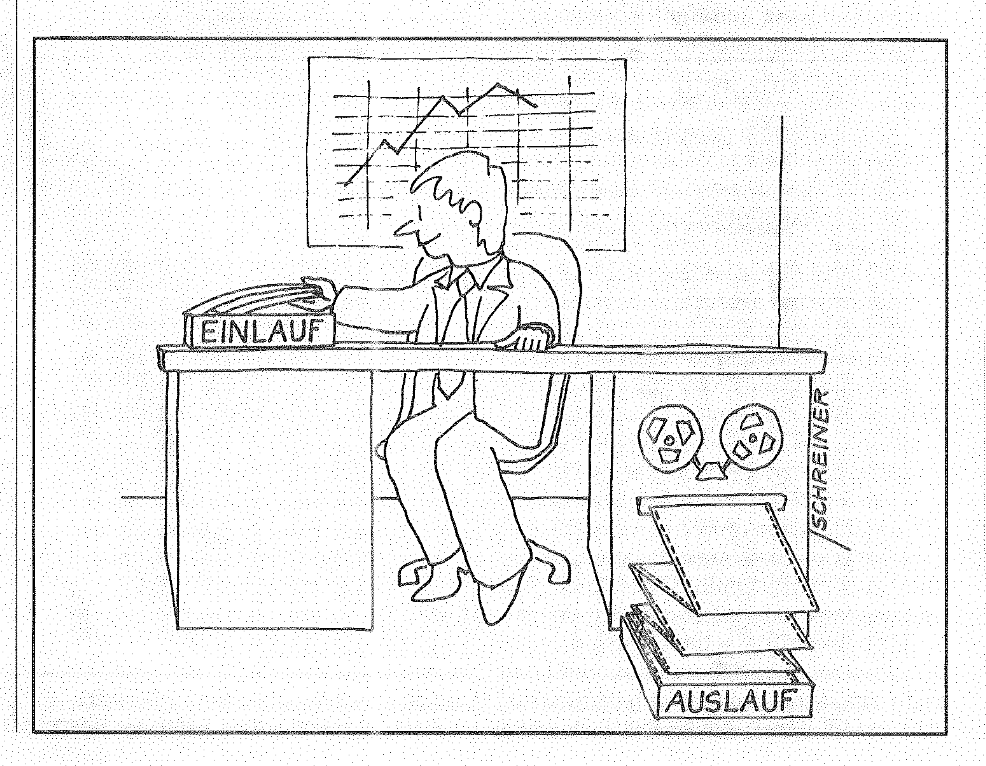
ten!). Dazu kommen Mini-Computer, zum Beispiel DECs LSI-11/PDP-11-Rechner. In jüngster Zeit häufen sich auch die Hinweise von Software-Anbietern, daß ihre jeweiligen Programme in UCSD-Pascal geschrieben seien (Byte).

Zur Sprache Pascal

UCSD-Pascal ist eine der modernsten und komfortabelsten Pascal-Versionen überhaupt. Beispielsweise hat es eine volle String-Verwaltung (Verketten, Löschen, Muster suchen, Teile kopieren usw.) für den zusätzlichen Standard-Typ string. Moderne Sprach-Erweiterung wie das exit-Statement (sauberer irregulärer Abbruch von Prozeduren, sog. strukturiertes goto), Segmentierung von Programmteilen (Overlays, d. h. Programmteile werden abwechselnd in den Arbeitsspeicher geladen) und vor allem das Unit-Konzept, mit dem man separate Moduln bildet (getrennte Compilierung), die globale und private Prozeduren sowie Datenobjekte besitzen, nehmen manche der Konzepte von Ada bereits vorweg und begeistern den Feinschmecker sowie den Praktiker. Hat man beispielsweise ein Plotterpaket geschrieben und als „UNIT" in der Systembibliothek abgelegt, so verwendet man im Benutzer-Programm nach „uses plotlib" die selbst definierten Plotter-Befehle, als wären sie Features der Sprache selbst!
Die Version IV.0 dringt bereits in das Gebiet der Parallel-Verarbeitung vor. Die rund 600 Seiten starke Dokumentation beschreibt alle Abweichungen von Standard-Pascal sowie den Gebrauch des Betriebssystems sehr detailliert und exakt. Der Autor konnte bis zum Zeitpunkt der Manuskripterstellung dieses Artikels nur mit der Version II.0 Erfahrungen sammeln, hat aber inzwischen von Softech (Adresse bei [2]). Angaben über das System IV.0 erhalten, das seit einigen Monaten angeboten wird. IV.0 vereint die Eigenschaften der früheren Version II.0, II.1 (Apple) und III.0 (Microengine) und ergänzt sie. Die Source-Codes der bisherigen Programme sind aufwärtskompatibel, sie können in IV.0 neu übersetzt werden, ohne daß man sie umschreiben muß. Der P-Code wurde allerdings verändert. Benutzer der bisherigen Systeme sollten daher rückfragen, zu welchen Bedingungen sie ein Updating erhalten. Damit ist leider auch ein kritischer Punkt aufgezeigt.
Das neue Design enthält streng professionell ausgerichtete Features und dürfte damit alle anderen gegenwärtig für Microcomputer erhältlichen Software-Systeme weit hinter sich lassen.

Was die Version IV.0 kann

☐ Es können größere Programme compiliert werden. Teile des Betriebssystems laden sich nach Bedarf automatisch in den Hauptspeicher und vermögen diesen besser auszunutzen.
☐ Der Anwender-Programmierer kann die Anwesenheit seiner Programm-Segmente (Overlays) im Speicher steuern – und so etwa Zeit oder Platz sparen.
☐ Die Sprache enthält jetzt die nötigen Grundoperationen für Multitasking (Concurrent processes, d. h. quasi-paralleler Ablauf mehrerer Programme).
☐ Diverse Einschränkungen hinsichtlich der Zahl von Code-Segmenten (Overlays) und beim Gebrauch von Units (Bibliotheks-Paketen) wurden weiter gespannt:
Ein Programm darf jetzt nahezu beliebig viele UNITs (Pakete bzw. Bibliotheken) und SEGMENTs (Overlays, die sich abwechselnd in den Speicher laden) verwenden. UNITs dürfen nunmehr selbst segmentiert werden. Ferner können sie privat (private) andere UNITs oder FILEs (Dateien) anfordern. Der Bibliotheks-Benutzer wird also nicht mehr mit einschlägigen Deklarationen, die er in seinem Programm vornehmen müßte, behelligt, er kann auch nicht versehentlich auf die geschützten Objekte der Bibliothek zugreifen. Daraus resultiert eine höhere Programmsicherheit. Schließlich ist nun der Weg offen, nahezu beliebig große Programmsysteme zu entwerfen, ungeachtet des begrenzten Speichers der 8-Bit-Micros.
☐ UNITs können nunmehr Initialisierungs- und Terminierungscode enthalten, der automatisch ausgeführt wird, wenn das Programm startet bzw. beendet wird. Damit entfällt die Notwendigkeit, beispielsweise bei Benutzung eines Graphikpaketes, in jedem Benutzerprogramm eine Initialisierungsanweisung für die Schnittstelle einzufügen.
☐ Es besteht die Möglichkeit, von verschiedenen Programmen aus auf dieselbe Code-Kopie einer UNIT auf der Diskette zuzugreifen. Damit verringert sich der Speicherbedarf für komplexe Anwendungen.
☐ Codefiles und Directories (Inhaltsverzeichnisse der Diskette) sind unabhängig von der Wort-Struktur der Maschine (Anordnung von HO- und LO-Byte).
☐ Der lange vermißte, aber angekündigte P-Code-Debugger ist endlich verfügbar.
☐ Programm-Chaining ist möglich, d. h. Programme können sich gegenseitig aufrufen (man kennt dies etwa von komfortableren Basic-Implementierungen).
☐ Die Standard-Ein- und Ausgabe-Files des Systems, über die es normalerweise (Terminal) angesprochen wird, können umdirigiert werden. Damit sind einfache Steuerfiles möglich, wie man sie etwa vom CP/M-Submit kennt. Ein Anwender muß also nicht das System starten und sein Programm aufrufen, das System kann dies selbst tun.
☐ Die Dokumentation soll nun sowohl systemorientiert als auch benutzerorientiert vorliegen (die II.0-Dokumentation konnte man bisher nicht unbedingt als benutzerfreundlich bezeichnen).



□ Weitere Cross-Assembler für den 6809 und Z-8 sind verfügbar.
□ Ein schneller und leistungsfähiger Basic-Compiler wird angeboten. Er erlaubt lange Variablen-Namen, optionale Zeilennummern, if-then-else-Strukturen, INCLUDE-Files (d. h. das Einfügen von getrennten Source-Texten nach Anweisung im Programm), virtuelle Arrays, getrennte Übersetzung von Basic-Unterprogrammen und ist P-Code-kompatibel mit dem Pascal und dem Fortran 77.
□ Pascal, Fortran, Basic sowie Assembler-Code können zusammengelinkt (verbunden) werden.
□ Implementierung für die Prozessoren 8086/8087 und 6800 sind angekündigt für Ende 1981.
Es wird auch angekündigt, daß künftig die Umwandlung von P-Code in den eigenen Maschinencode der jeweiligen Prozessoren möglich sein soll, um die endgültig ausgetesteten Programme noch schneller zu machen. Eine Benutzergruppe, der man zum Jahresbeitrag von 20 $ beitreten kann, gibt es unter der folgenden Adresse:
USUS, Chip Chain Secretary,
P.O. Box 1148, La Jolla,
Ca. 92038.

Literatur

[1] *Gilbreath, J.*: A High-Level Language Benchmark. Byte 1981, Heft 9, S. 180.
[2] Softech Microsystems: UCSD-Pascal. Users Manual. Softech Microsystems, 9494 Black Mountain Road, San Diego, CA 92126, USA.
[3] *Bowles, K.L.*: Microcomputer Problem Solving Using Pascal. Springer-Verlag, Berlin, Heidelberg, New York.
[4[*Bowles, K.L.*: Beginners Guide for the USCD-Pascal System. McGraw-Hill, New York.
[5] *Jensen, K., Wirth, N.*: Pascal User Manual and Report. Second edition. Springer-Verlag, Berlin, Heidelberg, New York.

Schnelle Stichwortsuche beim PET

Im Heft 22/1980 der FUNKSCHAU wurde ein Maschinenprogramm für den CBM vorgestellt, das einen für Suchvorgänge interessanten Befehl implementierte, die „INSTRING"-Funktion [3]. Diese neue Funktion gestattet in Verbindung mit dem in 1/1980 vorgestellten Editor eine schnelle Stichwortsuche [4]. Nachfolgend findet sich eine Version, die auf dem alten Betriebssystem des PET lauffähig ist (*Bild 1*).
Die Funktion „INSTRING" bietet auf elegante Weise die Möglichkeit, String-Arrays nach mehreren Schlüsselworten zu durchsuchen. Nur wenn alle Schlüsselworte im String enthalten sind, wird dieser ausgegeben.
Damit kann man z. B. ein Literaturverzeichnis all jener Veröffentlichungen erstellen, die sich mit einem bestimmten Thema befassen. Werden nur wenige Schlüsselworte angegeben, ist der Literaturnachweis möglicherweise sehr groß. Mit zunehmender und damit einschränkender Anzahl von Schlüsselworten schrumpft die Liste auf die wirklich das Problem treffenden Beiträge zusammen. Die Implementierung der Schlüsselwortsuche in das Texteditor-Programm zeigt der Programmausschnitt in *Bild 2*. Ohne Änderung sind damit bis zu 11 Schlüsselworte verarbeitbar. Da die Suche innerhalb eines Strings sofort abgebrochen wird, wenn eines der Schlüsselworte nicht vorkommt, erhöht sich die Suchzeit nur unwesentlich.
Das Anpassen des Maschinen-Programms an das Betriebssystem des PET war dank der im Sonderheft „Mikrocomputer-Anwendungen" abgedruckten Übersicht über die Betriebssystem-Routinen von PET und CBM ohne allzu große Probleme möglich [5]. *Peter Weber*

```
 270 J=-1:I=J:REM SUCHROUTINE
 280 PRINT"SUCHBEGRIFFE: (MIT 'END' ABSCHLIESSEN)
 290 I=I+1:INPUTB$(I):IFB$(I)<>"END"ANDI<10THEN290
 300 I=I-1
 310 J=J+1:IFA$(J)="END"THENPRINT:RETURN
 320 FORII=0TOI:S=@INST(A$(J),B$(II)):IFS=0THENII=I
 330 NEXT:IFS=0THEN310
 340 PRINTA$(J)
 350 GETE$:IFE$<>" "THEN310
 360 PRINT"▮ F=FORTSETZUNG,S=STOP▮"
 362 GETE$:IFE$<>"S"ANDE$<>"F"THEN362
 364 IFE$="F"THEN310
 366 RETURN
READY.
```

Bild 2. So erfolgt die Einbindung der neuen Suchfunktion in den Texteditor, den die FUNKSCHAU in Heft 22/1980 veröffentlichte

```
 100 PRINT"BITTE WARTEN
 110 X=7821:Y=PEEK(135)-32:X=256*Y+X
 120 FORI=0TO370
 130 READA:B=B+A:IFA<0THENA=Y-A
 140 POKE(X+I),A
 150 NEXTI
 160 IFB<>45619THENPRINT"▮▮▮▮ACHTUNG▮: EINGABE FEHLER":END
 170 SYS(7821+256*Y)
 180 PRINT"▮▮▮▮▮▮INSTRING-FUNKTION JETZT AKTIV
 190 REM
 1000 DATA 169,141,133,134,169,-30,133,135,169,076,133,203,169,162,133,204,169
 1010 DATA -30,133,205,096,201,064,240,036,201,058,176,247,076,207,000,230,201
 1020 DATA 208,002,230,202,096,165,201,208,002,198,202,198,201,096,032,173,-30
 1030 DATA 208,003,032,180,-30,160,000,177,201,096,164,202,192,002,240,214,169
 1040 DATA 000,133,001,133,193,032,194,-30,201,178,240,003,076,028,206,032,167
 1050 DATA 204,165,008,072,165,150,133,160,165,151,133,170,032,173,-30,160,005
 1060 DATA 032,194,000,217,242,-31,208,226,136,208,245,032,189,-30,032,040,206
 1070 DATA 032,169,204,032,223,-31,160,000,177,150,133,002,200,177,150,141,122
 1080 DATA -31,141,157,-31,200,177,150,141,123,-31,141,158,-31,032,017,206,032
 1090 DATA 040,206,032,169,204,032,223,-31,160,000,177,150,133,165,200,177,150
 1100 DATA 141,119,-31,141,152,-31,200,177,150,141,120,-31,141,153,-31,032,197
 1110 DATA -30,201,041,240,039,201,044,240,003,076,028,206,032,173,-30,032,184
 1120 DATA 204,032,167,204,032,171,208,165,180,133,001,197,002,144,002,176,060
 1130 DATA 032,197,-30,201,041,240,003,076,028,206,166,001,173,048,004,221,034
 1140 DATA 004,240,009,232,228,002,208,243,162,000,240,029,134,161,169,000,133
 1150 DATA 171,230,161,230,171,164,171,196,165,240,012,185,048,004,164,161,217
 1160 DATA 034,004,208,221,240,234,232,134,193,165,160,133,150,165,170,133,151
 1170 DATA 104,016,014,160,000,169,000,145,150,200,165,193,145,150,076,212,-31
 1180 DATA 169,000,133,177,164,193,132,178,162,144,056,032,027,219,166,150,164
 1190 DATA 151,032,166,218,104,104,104,104,032,173,-30,076,181,198,096,224,000
 1200 DATA 208,251,162,008,189,247,-31,032,210,255,202,208,247,160,040,076,123
 1210 DATA 245,040,084,083,078,073,071,078,073,082,084,083,063,013
```

Bild 1. Dieses Basic-Programm initialisiert ein Maschinenprogramm, das beim alten PET-2001-Betriebssystem das Suchen nach praktisch beliebig vielen Begriffen in Strings zuläßt

Literatur

[1] Basic-Texteditor, FUNKSCHAU 1980, Heft 1.
[2] Basic-Texteditor. FUNKSCHAU 1980, Heft 22.
[3] *Martin, Reinhold*: Schnelle Stichwortsuche beim CBM, FUNKSCHAU 1980, Heft 22.
[4] Schnelles Suchen beim Texteditor. FUNKSCHAU 1980, Heft 23.
[5] *Martin, Reinhold; Smode, Dieter*: ROM und RAM bei PET und CBM. Sonderheft „Mikrocomputer-Anwendungen", Franzis-Verlag.

Werner Lang

AIM-65 decodiert DCF 77

Das folgende Programm ermöglicht es, unter Verwendung eines geeigneten Empfängers für den Zeitzeichensender DCF 77 die ausgestrahlte amtliche Zeit auf dem Display des Mikrocomputers PC-100 bzw. AIM-65 zur Anzeige zu bringen. Ferner ist die Programmierung acht verschiedener Alarmzeiten möglich.

```
<↑>=0000 20 13 00 A9 00 85 F6 8D 03 A0 8D 0C A0 8D 0B A0
< > 0010 4C 29 00 A9 00 85 F9 85 FB 85 FC 85 FD 85 FE A9
< > 0020 99 85 FA A9 00 85 F8 60 A9 01 8D 0D A0 2C 0D A0
< > 0030 F0 FB A9 00 85 FD A9 80 8D 09 A0 A9 20 2C 0D A0
< > 0040 F0 FB E6 FD A5 FD C9 20 D0 EC A9 00 2C 01 A0 F0
< > 0050 D7 A9 01 8D 0D A0 2C 0D A0 F0 FB E6 F9 10 F8 A9
< > 0060 01 65 FA 85 FA D8 A5 F9 C9 01 D0 03 20 00 04 A5
< > 0070 F9 C9 15 30 03 4C 7E 00 20 3F 03 4C 51 00 A5 F9
< > 0080 C9 3B D0 06 20 3F 03 4C 29 02 20 3F 03 A9 00 85
< > 0090 FD A9 20 8D 09 A0 2C 0D A0 F0 FB E6 FD A5 FD C9
< > 00A0 06 D0 EE A9 80 2C 01 A0 F0 03 4C 5E 03 A9 00 85
< > 00B0 FD A9 20 8D 09 A0 2C 0D A0 F0 FB E6 FD A5 FD C9
< > 00C0 0C D0 EE A4 FC C0 04 30 04 A9 05 85 F8 A5 FB C5
< > 00D0 F8 F0 03 4C 00 02 4C 0F 02 00 FF 00 FF 00 FF FF
<↑>=0200 E6 FB AD 01 A0 29 80 05 FE 4A 85 FE 4C 51 00 AD
< > 0210 01 A0 29 80 05 FE 49 FF A6 FC 95 F0 E6 FC A9 00
< > 0220 85 FE A9 01 85 FB 4C 51 00 A9 00 85 FD A9 50 8D
< > 0230 08 A0 A9 C3 8D 09 A0 A9 20 2C 0D A0 F0 FB E6 FD
< > 0240 A9 14 C5 FD D0 E7 18 F8 A9 01 65 FA 85 FA D8 20
< > 0250 3F 03 A9 A0 8D 09 A0 A9 20 2C 0D A0 F0 FB A9 80
< > 0260 2C 01 A0 F0 0A A9 01 85 F6 20 13 00 4C 72 02 4C
< > 0270 5E 03 A5 F0 29 0F 8D CB 03 A5 F0 29 70 20 1E 03
< > 0280 8D CC 03 A5 F1 29 0F 8D CD 03 A5 F1 29 30 20 1E
< > 0290 03 8D CE 03 A5 F1 2A 2A 29 01 85 F5 A5 F2 0A 29
< > 02A0 0F 05 F5 8D CF 03 A5 F2 29 18 20 1F 03 8D D0 03
< > 02B0 A5 F2 29 E0 20 1D 03 8D D5 03 A5 F3 29 0F 8D D1
< > 02C0 03 A5 F3 29 10 20 1E 03 8D D2 03 A5 F4 29 08 85
< > 02D0 F5 A5 F3 20 1D 03 05 F5 8D D3 03 A5 F4 20 1E 03
< > 02E0 8D D4 03 A2 00 BD CB 03 09 30 9D CB 03 E8 E0 0A
< > 02F0 D0 F3 A0 00 C8 CC D5 03 D0 FA 98 0A A8 B9 A6 03
< > 0300 8D D5 03 C8 B9 A6 03 8D D6 03 A2 00 BD CB 03 BC
< > 0310 9C 03 99 D7 03 E8 E0 0C D0 F2 4C 51 00 4A 4A 4A
< > 0320 4A 4A 60 A9 01 25 F6 D0 01 60 A2 00 86 F7 BD D7
< > 0330 03 09 80 20 7B EF E6 F7 A6 F7 E0 12 D0 F0 60 A9
< > 0340 01 25 F6 D0 01 60 A5 FA 20 1E 03 09 B0 A2 12 20
< > 0350 7B EF A5 FA 29 0F 09 B0 A2 13 20 7B EF 60 20 13
< > 0360 EA A2 00 BD B6 03 20 7A E9 E8 E0 05 D0 F5 4C 00
< > 0370 00 20 13 EA A2 00 BD BB 03 20 7A E9 E8 E0 10 D0
< > 0380 F5 A9 2E 8D 09 03 8D DC 03 A9 3A 8D E5 03 8D E8

<↑>=0390 03 A9 20 8D DF 03 8D E2 03 4C 00 00 10 0F 0D 0C
< > 03A0 01 00 04 03 07 06 09 0A 4D 4F 44 49 4D 49 44 4F
< > 03B0 46 52 53 41 53 4F 45 52 52 4F 52 50 4C 45 41 53
< > 03C0 45 20 57 41 49 54 2E 2E 2E 2E 2E 00 FF 10 7F 00
<↑>=0400 20 23 03 A0 00 C8 C0 09 F0 25 B9 34 04 8D 13 04
< > 0410 A2 00 BD CE 04 DD D7 03 D0 EB E8 E0 11 D0 F3 A9
< > 0420 C0 8D 0B A0 A9 80 8D 02 A0 A9 04 8D 05 A0 60 A9
< > 0430 00 8D 0B A0 60 50 62 74 86 98 AA BC CE 8D FB F7
```

Bild 1. Hex-Listing des Programms. Die Prüfsummen der einzelnen Teile sind: FROM 0 TO D9 = 729F; FROM 200 TO 3CB = 38C3; FROM 400 TO 43D = 1C51 (vgl. mc 2/1981, S. 36)

Da sich die Funkempfänger für den Zeitzeichensender DCF 77 (77,5 kHz) immer größerer Beliebtheit erfreuen und man selbst von diesem Fieber nicht ganz verschont bleibt, liegt es nahe, deren Funktionen, soweit es sich um die digitale Signalverarbeitung handelt, von seinem Mikroprozessorsystem übernehmen zu lassen.

Im vorliegenden Fall ist das Programm (*Bild 1*) hierfür zwar für den AIM-65 geschrieben, es läßt sich aber ohne größere Schwierigkeiten auch auf andere 6502-Systeme umschreiben, insbesondere, wenn sie einen User-VIA 6522 besitzen, da bis auf die Interruptflag-Abfrage, die Anzeige- und Timerroutinen keine speziellen betriebssystemspezifischen Programmteile benutzt werden. Die wenigen genannten kann man sich z. B. mit Hilfe der in [1] tabellierten Monitor-Unterprogramme an sein System anpassen.

Der Empfänger für 77,5 kHz

Der AIM-65 mag zwar Erstaunliches leisten, aber ganz ohne einen geeigneten Empfänger, der das amplitudenmodulierte Langwellensignal demoduliert und mit TTL-Pegel bereitstellt, geht es nicht. Doch sollte man sich von dieser kleinen Hürde nicht davon abhalten lassen, das Programm in Betrieb zu nehmen. Wenn man nicht allzu hohe Anforderungen an die Zuverlässigkeit stellt, läßt sich solch ein Empfänger mit nur geringem Aufwand aufbauen. So z. B. verwendet der Verfasser mit Erfolg eine Schaltung nach eigenem Entwurf, die sich für ca. 15 DM Bauelementekosten realisieren läßt. Für eine hohe Betriebssicherheit sollte man aber schon einen Empfänger mit Quarzfilter vorsehen, da sich sonst vor allem Probleme mit der 5. Harmonischen der Fernsehzeilenfrequenz ergeben können [2]. Wer den Selbstbau scheut, kann einen entsprechenden Schaltungsaufbau auch erwerben. Solch ein Empfänger wird z. B. von der Firma Völkner (Braunschweig) angeboten, bei dem man allerdings noch einen Demodulator nachschalten muß. Wichtig ist, daß das vom Empfänger abgegebene Digitalsignal der Hüllkurve

des Sendesignals entspricht. Der Sekundenbeginn ist folglich durch eine abfallende Flanke und die Trägerabsenkung durch den TTL-Pegel Low gekennzeichnet. Ist dies bei dem verwendeten Empfänger umgekehrt, kann man zwar auch durch ein paar kleine Veränderungen im Programm zu Erfolg kommen, einfacher und schneller dürfte aber eine Invertierung des Signals z. B. durch ein 7400-Gatter sein.

Das Programm in 6502-Maschinensprache

Die vom Zeitzeichensender seriell im BCD-Code ausgestrahlte Information wird einschließlich der Prüfbits innerhalb der 21. und 58. Sekunde eines Minutendurchgangs übermittelt. Dabei handelt es sich jeweils um die Daten der nächstfolgenden Minute. Somit hat man, ganz gleich ob z. B. ein Schaltjahr oder der Wechsel zur Sommerzeit vorliegt, immer die exakte Zeit.
Nachdem durch die fehlende Trägerabsenkung zur 59. Sekunde der bevorstehende Minutenwechsel erkannt wurde, wird ab der 21. Sekunde jeweils 150 ms nach der negativen Flanke der Port PA7 abgefragt und dessen Zustand dem Zwischenspeicher FE übergeben. Erst nachdem dieser nach acht Abfragen aufgefüllt ist, wird sein Inhalt in einem weiteren Speicher ab F0 abgelegt, um die nachfolgenden Datenbits aufnehmen zu können. Die Decodierung der formlos aneinandergereiht abgespeicherten Einlesewerte vom BCD-Code in die zur Anzeige notwendige Form wird vor jedem Minutenwechsel mittels einer Maskenabfrage und einer einfachen Umcodierungsvorschrift in den ASCII-Code vorgenommen. Sodann werden die Zeichen durch eine Umordnungstabelle in das gewünschte Anzeigeformat gebracht und dem Auslesespeicher übergeben, von wo sie zu Beginn der neuen Minute das Display übernimmt.
Die Anzeige der Sekunden geschieht durch dezimales Hochzählen nach jeder negativen Flanke und entsprechender Umcodierung in das ASCII-Format. Lediglich nach der 58. Sekunde muß ein Timer gestartet werden, der die 59., nicht ausgesendete Sekunde festlegt. Zu jeder neuen Minute wird der Speicherbereich für die Alarmzeiten abgefragt und abhängig vom Vergleichsergebnis für die Dauer einer Minute ein Alarmton erzeugt.
Zur Fehlererkennung von gestörten Signalen ist während der Dateneinlesephase jeweils etwa 50 ms nach der abfallenden Flanke eine Abfrage des Ports vorgesehen. Ist der Signalzustand High, muß ein Fehler vorliegen. Ein durch eine Störung ausgebliebener Sekundenimpuls wird spätestens nach einer Minute erkannt, wenn nämlich die dann nicht vorhandene Synchronität zwischen dem tatsächlichen und dem vermeintlichen Minutenwechsel wiederum durch eine Portabfrage hervorgeht. In beiden Fällen gibt AIM-65 eine Fehlermeldung aus und wartet auf einen neuen Minutenanfang, um seinen Einlesezyklus fortzusetzen.
Der Verfasser hat durch eine Reihe von simulierten Störungen ermittelt, daß durch diese Fehlererkennungsmethode ein unkorrektes Empfangssignal mit großer Wahrscheinlichkeit erkannt wird. Auf eine Auswertung der Prüfbits wurde bewußt verzichtet, da gestörte Signale hierdurch nur mit einer relativ geringen Sicherheit zu analysieren wären.
Anwender, die nur die 1-KByte-Version des AIM besitzen, müssen auf die Möglichkeit der Alarmzeiteingabe verzichten. Für sie endet das Programmlisting bei 03CA. Es ist lediglich bei 006C die Unterprogrammadresse von 0400 auf 0323 zu ändern.
Das komplette Programm benötigt außer den aus dem Hexlisting hervorgehenden Speicherplätzen die RAM-Bereiche:
00F0...00FE
03CB...03E8
0450...04E0

Anschluß und Bedienung der Alarmeinrichtung

Das vom Empfänger kommende TTL-Signal wird auf Port PA7 (J1, Pin 8) geführt. Zudem ist PA7 mit CA2 (J1, Pin 21) zu verbinden.
Zur Erzeugung des Alarmtons ist ein Kleinlautsprecher über PB7 (J1, Pin 15) anzusteuern (*Bild 2*). Zu empfehlen ist ein Schalter, mit dem er sich vorzeitig abschalten läßt.
Nach dem Start mit F1 bei 0371 bittet der AIM 65, zu warten, denn er braucht wie alle Funkuhren zunächst etwa zwei Minuten Zeit, um die gesendeten Daten einzulesen, und meldet sich dann prompt mit der empfangenen Zeitinformation.
Falls von der Möglichkeit des Alarmtons Gebrauch gemacht werden soll, sind die entsprechenden Zeiten mit dem Text-Editor vor dem Start ab 0450 nacheinander zeilenweise einzugeben. Dabei ist streng darauf zu achten, daß das Eingabeformat mit dem Anzeigeformat genau übereinstimmt. Auch die beiden Leerplätze vor und nach dem Wochentag sind einzutippen. Lediglich nach dem Minuten-Einer folgt bereits das Zeilenendzeichen Return, da die Alarmzeiten nur minutengenau zu programmieren sind. Ein Beispiel soll dies verdeutlichen: 22.05.81 FR 19:35...
Die zeitliche Ordnung ist dabei uninteressant; ebenso die Anzahl der Alarmzeiten. Sind mehr als acht Zeiten eingegeben, werden die restlichen einfach ignoriert.

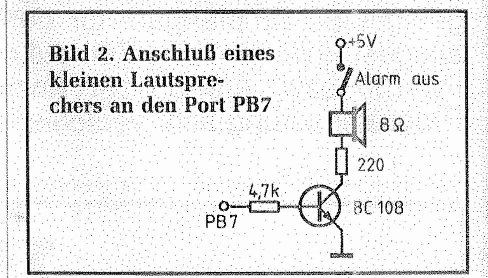


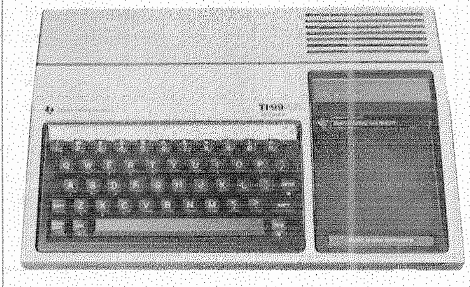
Bild 2. Anschluß eines kleinen Lautsprechers an den Port PB7

Literatur

[1] ROM und RAM in KIM und AIM, in: „Mikrocomputer-Anwendungen", Franzis-Sonderheft Nr. 33.
[2] Mikrocomputergesteuerte Funkuhr im Mini-Format. FUNKSCHAU 1979, Heft 14, S. 839.

Video-Computer-System

Texas Instruments hat seinen Computer TI99/4 aufgemöbelt und eine Version A daraus gemacht. Hervorstechendstes Merkmal dieses neuen Computersystems ist die Schreibmaschinentastatur. Daran ist jetzt auch ein größeres Programmierprojekt ermüdungsfrei zu erledigen. Innen sitzt der bewährte Prozessor 9900. In der Grundversion wird der Computer mit 16 KByte RAM, 26 KByte ROM mit Betriebssystem, Interpreter für die Grafiksprache GPL, 4 KByte Basic-Interpreter (TI-Basic-ANSI-Min), 4 KByte Monitor für 9900-Maschinensprache geliefert. Der Video-Ausgang liefert jetzt ein PAL-Signal, also das Signal, das jeder Farbfernseher an seiner AV-Buchse in unserem Land erwartet. Ro.



Der neue Texas-Computer mit Schreibmaschinentastatur

Horst W. Albrecht, Christian Spitschka

EDV in der Schulverwaltung

Für Schulen waren bisher EDV-Anlagen oft zu teuer und wurden daher nur in begrenzter Zahl angeschafft. Seit jedoch die ersten Kleincomputer erschienen und die Anschaffungskosten für eine Anlage unter 15 000 DM sanken, kann man an den Einsatz der EDV-Technik in der Schulverwaltung denken.

Daß Schulleitungen und Sekretariate durch ständig zunehmende administrative und verwaltungstechnische Arbeit immer stärker belastet werden, wird sicherlich von niemandem bestritten. Es kann aber nicht Aufgabe von Schulleitern, Konrektoren und Fachlehrern sein, sich oft auch nach der Dienstzeit an die Maschine setzen zu müssen, um statistische Meldungen für vorgeordnete Dienststellen zu tippen oder sonstige Terminarbeiten erledigen zu müssen, zu denen das Sekretariat wegen Arbeitsüberlastung nicht gekommen ist.
Für solche Arbeiten sind hochqualifizierte Fachleute und erfahrene Pädagogen, ganz nüchtern betrachtet, einfach zu wertvoll.
Sind Personallösungen zur Verstärkung der Sekretariate aufgrund der angespannten Haushaltslage jedoch nicht denkbar, sollte man an den Einsatz anderer Arbeitsmethoden, zum Beispiel EDV denken.
Kleincomputer mit 32 KByte Hauptspeicher, Floppy und Drucker sind heute schon für ca. 9000 bis 12 000 DM zu bekommen. Das dürfte etwa so viel sein, wie eine Halbtagskraft für das Sekretariat einer Volks- oder weiterführenden Schule mit allen Nebenkosten in acht Monaten kosten würde.
Aber auch der beste, kommerziell vielseitig verwendbare Mikrocomputer nützt der Schulverwaltung wenig, wenn es an entsprechender Software mangelt.

Welche Anforderungen sollte ein Computerprogramm erfüllen?

Als einige der wesentlichen Voraussetzungen seien genannt:

- Es muß möglichst flexibel und variabel sein, d. h. es sollte sich für den Einsatz in den verschiedenen Schulbereichen eignen.
- Es sollte so programmiert sein, daß es nach Möglichkeit auf verschiedenen Gerätetypen eines Fabrikates verwendet werden kann und sich ggf. auch auf andere Computersysteme umschreiben läßt.
- Ein Verwaltungsprogramm muß bedienerfreundlich sein, um die oft noch vorhandene Scheu vor dem „Unbekannten", das sich „EDV" nennt, abbauen zu helfen.
- Wichtig sind daher verständliche Eingaben. Ein Programm darf nicht etwa durch vielstellige Schlüsselzahlen undurchschaubar werden.

Im folgenden möchten wir an einem konkreten Beispiel die Punkte zusammenfassen, die nach unserer Meinung zu einem Schulverwaltungsprogramm gehören; wobei zu bedenken wäre, daß der Computer dazu dienen soll, die Masse der zeit- und arbeitsaufwendigen Routinearbeit zu erledigen.

Schwerpunkte des Programms „SVP" für die Schulverwaltung

Das Programm muß auf die individuellen Wünsche der jeweiligen Schule eingehen können.
Auch ohne besondere Programmiererfahrung muß das Programm in kurzer Zeit auf diese individuellen Wünsche zugeschnitten werden können.
In einer Vorbereitungsroutine müssen einzugeben sein:

1. Der Schulname.
2. Die Anzahl der Klassen, die – z. B. beim Erstellen einer Statistik – auf einmal verarbeitet werden sollen.
3. Die Maximalbelegung von Schülern in einer Klasse.
4. Die Zahl der Schülermerkmale.
5. Die Schülermerkmale selbst (15 Merkmale sind z. B. für statistische und Zeugnisroutinen schon gesetzt). Alle Schülermerkmale darüber hinaus können von der Schule beliebig gewählt werden.

Formate von Klassenlisten müssen frei wählbar sein. In speziellen Routinen können besondere Wünsche ohne Schwierigkeiten verwirklicht werden.
Einmal eingegebene Daten können jederzeit und schnell auf dem Bildschirm oder durch den Drucker sichtbar gemacht werden.
(Es gibt auch eine Routine, mit welcher sämtliche Dateien von Klassen auf dem Bildschirm „durchblättert" werden können.)
Alle eingegebenen Formate und Schülerdaten müssen jederzeit geändert, gelesen oder gelöscht werden können.
Das Programm muß dem Benutzer immer „sagen", was er machen soll, in welchem Programmteil er sich befindet und wie er notfalls wieder auf die Ausgangsroutine zurückkommt.
Es müssen von mehreren Klassen zusammen die Statistiken zu erstellen sein.
Das Programm soll ein echtes Hilfsmittel für die Schulverwaltung sein und nicht etwa die Sekretärin in eine Abhängigkeit vom Computer geraten lassen.
Es soll den Verwaltungskräften die mühsame Routinearbeit abnehmen.
Berücksichtigt ein Schulverwaltungsprogramm alle aufgezeichneten Gesichtspunkte, so ergibt sich daraus ein Programmangebot mit folgenden Leistungsbereichen:

Programmangebot eines Schulverwaltungsprogrammes

Vorbereitungsroutinen
Einrichten einer Merkmaldatei; Lesen einer Merkmaldatei; Löschen einer Merkmaldatei.
Diskettenroutinen
Formatieren einer Diskette; Initialisieren einer Diskette.
Speicherungsprogramme
Klassenweises Erfassen mit beliebig vielen Merkmalen.
Erweiterungsprogramme
Leseprogramme
Zum Lesen der Daten eines Schülers; zum Erstellen einer Klassenliste am Bildschirm; zum Lesen aller Daten einer Klasse.
Druckprogramme
Klassenlistendruckprogramme; Eingabe von Listenformaten; Lesen und Ändern

von Listenformaten; Drucken von Klassenlisten; Erweitern der Datei für diverse Listenformate. Drucken der Daten eines Schülers; Drucken der Daten einer Klasse; Drucken von Adressen: der Eltern; der Schüler; der Schüler bei Volljährigkeit, ansonsten die der Eltern. Statistikdruckprogramm: Ausdrucken der Schüler mit und ohne Adressen; Suchen nach einem bestimmten Schülermerkmal; Suchen nach einem Jahrgang; Suchen nach der Volljährigkeit von Schülern; Ausdruck der Anzahl der Schüler mit einem bestimmten Merkmal. Dazu Möglichkeit der Namensausgabe. Drukken von Schulbescheinigungen: einer ganzen Klasse; eines Schülers. Papierjustierroutine für den Drucker.

Änderungsprogramme
Ändern von Daten; Ändern von Klassennamen (Vorrücken in die nächsthöhere Klasse).
Zugriffsroutine für Zeugniserstellungsprogramm (ZEP)
Leseroutine für das Klassenverzeichnis
Umbesetzungsprogramme
Löschprogramme: zum Löschen einzelner Schüler; zum Löschen von Klassen.
Wiederholerprogramm.
Duplizierprogramme
Von Diskette auf Kassette (eine Klasse);
von Diskette auf Diskette (eine Klasse);
von Kassette auf Kassette (eine Klasse);
von Diskette auf Diskette (die ganze Diskette);
von Kassette auf Diskette (eine Klasse).
Zur Ergänzung und Abrundung des Schulverwaltungsprogrammes (SVP) gehörte aber auch ein Zeugniserstellungsprogramm (ZEP), wobei es jeder Schule jedoch vorbehalten bleiben muß, ob sie dasselbe einsetzen möchte oder nicht. Es wäre sinnvoll, aber nicht obligatorisch notwendig.
Für Schulen, die an einem ergänzenden Zeugniserstellungsprogramm interessiert sind, wird nachstehend erläutert, welche Programmteile ein ZEP enthalten kann:
Errichten einer Fächerdatei
Einrichten einer Fächerdatei;
Lesen einer Fächerdatei;
Ändern der Fächerdatei;
Ausdrucken der Fächerdatei.
Einrichten einer Zeugnisdatei
Eingabe von Zeugnisnoten;
Lesen von Zeugnisnoten;
Ändern von Zeugnisnoten.
Eingabe von Zeugnisformaten
Drucken einer Hilfsmatrix;
Eingabe der Zeugnisformate;
Ausgabe eines Kontrollausdruckes;
Löschen von Zeugnisformaten.
Drucken von Zeugnissen
Drucken von Zeugnissen ganzer Klassen;
Drucken von Zeugnissen einzelner Schüler.
Mit diesen in Programmangebot und Leistungsspektrum beschriebenen Computerprogrammen „SVP-F (Schulverwaltung)" und „ZEP-F (Zeugniserstellung)" können erhebliche Teile administrativer Routinearbeit maschinell durchgeführt werden und damit sehr wesentlich zur Entlastung von Schulleitungen und Sekretariaten beitragen.

Alles in einem

Aus japanischer Entwicklung stammt ein sehr interessanter Tischcomputer. Das von OKI gebaute und in den USA auch von BMC vertriebene Gerät IF-800 (*Bild*) enthält alles in einem Gehäuse – Farbbildschirm, Tastatur mit getrenntem numerischem Feld, Doppel-Floppy-Laufwerk und sogar Drucker.
Das Display läßt eine Darstellung in unterschiedlichen Formaten programmgesteuert zu: 80/25, 80/20, 40/25 und 40/20 Zeichen sind möglich. Der eingebaute Matrix-Drucker arbeitet mit Normalpapier und beherrscht Groß- und Kleinschreibung. Jedes der beiden Floppy-Laufwerke kann 280 KByte speichern. Der Computer ähnelt im Konzept auffallend dem vor rund zwei Jahren von Siemens entwickelten PC-1000, der leider nie in Serie ging. Wann der IF-800 in Europa erhältlich sein wird, steht bisher noch nicht fest.
Außer mit dem Basic-Interpreter kann der Benutzer des IF-800 mit einem Basic-, Fortran- oder Cobol-Compiler arbeiten; ebenso ist eine Assemblerprogrammierung sowie die Benutzung des CP/M-Betriebssystems möglich. Das Gerät enthält einen Z-80-Mikroprozessor und 64 KByte RAM. Eine Erweiterung ist auch mit steckbaren ROM-Kassetten möglich.



Farbbildschirm, Drucker und zwei Minifloppy-Laufwerke sind im IF-800 von OKI/BMC bereits eingebaut. Die Grafik-Auflösung des Schirms beträgt 640 × 200 Punkte

Harald Krake

Text-I/O für Z80-Systeme

Angeregt wurde der Autor zu diesem Beitrag durch zahlreiche Leser, die Schwierigkeiten mit der Implementierung des in mc 1981, Heft 2 veröffentlichten Z80-Texteditors hatten. Im folgenden werden daher einige I/O-Routinen beschrieben, die sich auch grundsätzlich zur Anpassung systemfremder Maschinensoftware eignen.

```
;       I/O-Routinen zur Anpassung des Z80-Texteditors an TRS80 :
;       ------------------------------------------------------------
;       ( laeuft prinzipiell auf jedem System mit Videospeicher )
;
INCH    EQU   #002B              ;Tastaturroutine (s.Text)
OUTCH   EQU   #0033              ;UP druckt Zeichen auf Bildschirm
CURSOR  EQU   #4020              ;enthaelt aktuelle Cursoradresse
CR      EQU   #0D                ;ASCII fuer "carriage return"
;
        ORG   #4A00              ;beliebig
;
OUTPUT  PUSH  IX                 DD E5
        PUSH  DE                 D5          ;bei TRS80 noetig !
        CALL  OUTCH              CD 33 00    ;Zeichen ausgeben
        POP   DE                 D1
        POP   IX                 DD E1
        RET                      C9
;
INPUT   PUSH  IX                 DD E5       ; man weiss nie .....
INKEY   CALL  INCH               CD 2B 00    ;Tastaturabfrage
        AND   A                  A7          ;Taste gedrueckt ?
        JR    Z,INKEY            28 FA       ;Z=0, wenn ja
        CALL  OUTCH              CD 33 00    ;Zeichen drucken
        CP    CR                 FE 0D       ;"carriage return" ?
        JR    NZ,INKEY           20 F3       ;naechstes Zeichen
        LD    A,#20              3E 20       ;A ← "blank"
        LD    HL,(CURSOR)        2A 20 40    ;HL ← Cursoradresse
FCHAR   DEC   HL                 2B
        CP    (HL)               BE          ;"blank" ?
        JR    Z,FCHAR            28 FC       ;Z=0, wenn nicht
        INC   HL                 23          ;(HL)=Zeilenende + 1
        EX    HL,DE              EB          ;DE ← BUFEND
        POP   IX                 DD E1
        RET                      C9
;
; bei anderen Systemen muessen die PUSH & POP - Befehle eventuell
; geaendert werden. Ausserdem kann die Tastaturabfrage variieren.
;
```

Bild 1. I/O-Routinen zur Anpassung des Texteditors aus Heft 2/1981 an den Computer TRS-80

```
    **************** S C R E D I T ****************
    ein universeller Screen-Editor fuer Z80-Systeme
;
;
CURS    EQU   #CCCC          ;Adresse des virtuellen Cursors
              ;sollte nur von SCREDIT selbst belegt werden.
SBEG    EQU   #BBBB          ;Beginn des virtuellen Video-RAMs
        ;entspricht der linken, oberen Ecke des Bildschirms .
SEND    EQU   #EEEE          ;Ende+1 des virtuellen Video-RAMs
        ;entspricht der rechten, unteren Ecke des Schirms + 1
SLEN    EQU   #DDDD          ;Laenge des Video-RAMs = SEND-SBEG
LLEN    EQU   #AA            ;Anzahl der Zeichen in einer Zeile
SBL     EQU   #FFFF          ;Beginn der letzten, unteren Zeile
SSL     EQU   #2222          ;Beginn der zweiten Zeile=SBEG+SLEN
;
```

Bild 2. Das ist das Assembler-Listing eines Bildschirm-Verwaltungsprogramms für Z80-Systeme – da dieses Programm voll verschieblich ist, müssen lediglich die Parameter für den Simultan-Puffer dem jeweiligen Bildschirmformat angepaßt werden

Eigentlich ein Kuriosum: Entgegen allen Gepflogenheiten wird einmal ein Programm veröffentlicht, welches sich von der sonst gewohnten „ready-to-use-software" u. a. dadurch unterscheidet, daß es ohne Änderung am Maschinencode in praktisch jedes Z80-System implementiert werden kann, und dann entpuppt sich gerade dieses im guten Glauben als Vereinfachung gedachte Prinzip als größtes Hindernis. Grund genug, um auf die unterschiedlichen Möglichkeiten der I/O-Anpassung etwas näher einzugehen.

Unentbehrlich: der Textbuffer

Dialogorientierte Programme, wie z.B. ein Editor, benötigen einen Textbuffer, in dem sämtliche Eingaben vom Benutzer sowie i. d. R. auch gleichzeitig die Ausgaben des Programmes zwischengespeichert werden können und damit jederzeit als ganze Datensätze verfügbar sind. Bei den meisten Mikrocomputern (TRS80, Sharp, Nascom...) wird der Textbuffer gleichzeitig als Videospeicher verwendet und über eine entsprechende Elektronik auf einem Bildschirm abgebildet. Andere Systeme arbeiten jedoch nach dem Terminal-Prinzip (z. B. Elzet 80). Hier bilden der Bildschirmspeicher, die TV-Elektronik sowie i. d. R. auch das Tastaturinterface eine externe Einheit, die mit der CPU lediglich über eine I/O-Schnittstelle (z. B. 8-Bit-Parallelport) in Verbindung steht. Bei weniger komfortablen Systemen hat die CPU dann keine Möglichkeit, nachträglich einzelne Zeilen in den Arbeitsspeicher einzulesen. In diesem Fall kommt man leider nicht umhin, einen „Simultan-Buffer" durch einen Screen-Editor erzeugen zu lassen, der sich jede einzelne gedrückte Taste „merkt" und ein genaues Abbild des Bildschirminhaltes im RAM erstellt. Dies setzt natürlich voraus, daß sämtliche Steuerzeichen, die die externe Video-Logik interpretieren kann, auch vom Screen-Editor in exakt gleicher Weise verarbeitet werden. Da die Implementierung systemfremder Software auf einem solchen Computer u. U. sehr aufwendig sein kann, wird neben dem aus Bild 1 ersichtlichen Beispiel, wie man den in mc 1981, Heft 2, veröffentlichten Texteditor an den TRS80 anpaßt, ein universeller Screen-Editor beschrieben, der sich für nahezu jeden Z80-Computer eignet. Außerdem wird gezeigt, wie man dem etwas spartanisch ausgefallenen Betriebssystem des Nascom-1 einen für die Textverarbeitung erforderlichen „gehobenen Umgangston" beibringt.

Implementierung auf Systemen mit Videospeicher

Obwohl die in Bild 1 beschriebenen I/O-Routinen für den TRS80 geschrieben sind, eignen sie sich jedoch zumindest prinzipiell für jeden Z80-Computer mit Videospeicher. Es müssen dann lediglich die Adressen der Systemroutinen und eventuell die Maschinenbefehle zur Abfrage der Tastatur geändert werden. Beim TRS80 ist zu beachten, daß vor dem Aufruf des Texteditors der Cursor durch 3E 0E CD 33 00 eingeschaltet werden sollte, da man sonst „im Dunkeln tappt". Die beiden Systemroutinen haben folgende Funktion: Die OUTCH-Routine schreibt ein im Akku befindliches ASCII an die Stelle des aktuellen Cursors und interpretiert einige Steuerzeichen. Dieses Unterprogramm ist somit einem Screen-Editor vergleichbar. Die INCH-Routine fragt die Tastatur ab, wobei der Akku den ASCII-Wert der gedrückten Taste erhält. Wurde im Moment der Abfrage keine Taste gedrückt, dann enthält der Akku den Wert 00. Da *Bild 1* das komplette Assemblerlisting zeigt, dürfte es keine Schwierigkeiten bereiten, die Anpassung an andere Computer, wie z. B. MZ80K o. ä. vorzunehmen, denn oben genannte Systemroutinen sind in zumindest ähnlicher Form bei jedem Computer vorhanden.

Implementierung auf „Terminal-Systemen"

Da das Betriebssystem derartiger Computer dem Anwender oftmals keinen separaten Screen-Editor zur Verfügung stellt, muß dieser nachträglich implementiert werden. Der Texteditor hat dann wahlfreien Zugriff zu jeder beliebigen Bildschirmzeile, indem man einfach den Cursor auf das gewünschte Zeilenende stellt und „CR" drückt. Zum Nachrüsten des eigenen Betriebssystems zeigt *Bild 2* das komplette Assemblerlisting eines universellen Screen-Editors, der den unterschiedlichsten Bedürfnissen angepaßt werden kann. SCREDIT beherrscht zwar die wichtigsten, über allgemein übliche ASCII-Codes definierten Steuerzeichen, doch oft gilt für ein und dieselbe Steuerfunktion der altbekannte Lehrsatz:
„Zwei verschiedene Hersteller – drei verschiedene ASCII-Codes." Hinzu kommen bei einigen Betriebssystemen nicht selten etwas exotisch anmutende Funktionen, wie z. B.: „falls Cursor nicht am Zeilenanfang steht, dann führe CR+LF aus" oder „LF mit Löschen der untersten Zeile". Die wenigsten Leser werden da-

```
;       Steuerzeichen :
;
CLEFT   EQU     #08             ;Cursor um eine Stelle nach links
CRIGHT  EQU     #09             ;Cursor um eine Stelle nach rechts
CDOWN   EQU     #0A             ;line feed ,       .... nach unten
CUP     EQU     #0B             ;Cursor um eine Stelle nach oben
CHCLS   EQU     #0C             ;Cursor home & clear screen
CR      EQU     #0D             ;carriage return
BS      EQU     #18             ;Backspace & delete
CH      EQU     #1C             ;Cursor home = Ecke links oben
CBL     EQU     #1D             ;Cursor zurueck zum Zeilenanfang
;
        ORG     #0000           ;beliebig, da SCREDIT relocatable
;
SCREDI  AND     A               A7
        RET     Z               C8                  ;return if accu=00
        PUSH    AF              F5                  ;saemtliche
        PUSH    BC              C5                  ;primaer
        PUSH    DE              D5                  ;Register
        PUSH    HL              E5                  ;retten.
        LD      HL,(CURS)       2A CC CC            ;HL ← Cursoradresse
        CP      CRIGHT          FE 09               ;"cright" ?
        JR      Z,NEXT          28 05
        CP      #20             FE 20               ;Steuerzeichen ?
        JR      C,PLEFT         38 28               ;C=1, wenn St.zeichen
        LD      (HL),A          77                  ;sonst Zeichen drucken
NEXT    INC     HL              23                  ;Cursor um 1 n. rechts
        LD      (CURS),HL       22 CC CC            ;neue CURS-Adresse
        EX      HL,DE           EB
        LD      HL,SEND         21 EE EE            ;HL ← Bufferende+1
        SBC     HL,DE           ED 52               ;Cursor am Schirmende?
        JR      NZ,CONT         20 1A               ;Z=0, wenn nicht
        LD      HL,SBL          21 FF FF            ;HL ← neuen Z.-Anfang
        LD      (CURS),HL       22 CC CC            ;Cursor n. links unten
SCROLL  PUSH    HL              E5
        LD      DE,SBEG         11 BB BB            ;DE ← links oben
        LD      HL,SSL          21 22 22            ;HL ← zweite Zeile
        LD      BC,SLEN         01 DD DD            ;BC ← Bufferlaenge
        LDIR                    ED B0               ;scroll
        POP     HL              E1                  ;HL ← links unten
        LD      B,LLEN          06 AA               ;B ← Zeilenlaenge
CLEAR   LD      (HL),#20        36 20               ;untere
        INC     HL              23                  ;Zeile
        DJNZ    CLEAR           10 FB               ;loeschen.
CONT    XOR     A               AF                  ;A ← 00
;
; "Strickleiter" fuer die Routinen der einzelnen Steuercodes :
;
                                                    ;HL=Cursorposition !
PLEFT   CP      CLEFT           FE 08               ;"cleft" ?
        JR      NZ,PDOWN        20 0C
IFBEG   LD      DE,SBEG         11 BB BB
        EX      HL,DE           EB                  ;HL ← Bufferanfang
        SBC     HL,DE           ED 52               ;Cursor links oben ?
        JR      NC,PDOWN        30 04               ;C=0, wenn ja
        DEC     DE              1B                  ;Cursor um eine
        EX      HL,DE           EB                  ;Stelle nach links
        JR      NCRS            18 57
PDOWN   CP      CDOWN           FE 0A               ;"line feed" ?
        JR      NZ,PUP          20 13
        LD      DE,LLEN         11 AA 00            ;D ← 00 ; E ← LLEN
        EX      HL,DE           EB
        ADD     HL,DE           19                  ;Curs. um 1 n. unten
        PUSH    HL              E5
        LD      BC,SEND         01 EE EE            ;BC ← Bildschirmende+1
        SBC     HL,BC           ED 42               ;scroll noetig ?
        POP     HL              E1
        JR      C,NCRS          38 45               ;C=1, wenn nicht
        LD      HL,SBL          21 FF FF            ;HL ← Ecke links unten
        JR      SCROLL          18 C4               ;scrollen
PUP     CP      CUP             FE 0B               ;"cup" ?
        JR      NZ,PHCLS        20 09
        INC     HL              23
        LD      DE,LLEN         11 AA 00            ;D ← 00 ; E ← LLEN
        SBC     HL,DE           ED 52               ;eine Zeile nach oben
        XOR     A               AF                  ;nur bei Buffer in page 0
        JR      IFBEG           18 D0               ;oberste Zeile ?
PHCLS   CP      CHCLS           FE 0C               ;"clear screen" ?
        JR      NZ,PCR          20 0F
        LD      HL,SBEG         21 BB BB            ;HL ← Ecke links oben
        LD      DE,HL           54 5D               ;DE ← HL
        INC     DE              13                  ;DE ← zweites Zeichen
        LD      (HL),#20        36 20               ;1. Zeichen loeschen
        LD      BC,SLEN         01 DD DD            ;BC ← Bufferlaenge
        LDIR                    ED B0               ;Video-RAM loeschen
```

```
        LD      A,CH          3E 1C         ;"cursor home" ausf.
PCR     CP      CR            FE 0D         ;"carriage return" ?
        JR      NZ,PBS        20 0A
        LD      A,CDOWN       3E 0A         ;"line feed"
        CALL    SCREDI        CD 00 00      ;ausfuehren .
        LD      A,CBL         3E 1D         ;"cbl" ausfuehren
PBS     CP      BS            FE 18         ;"backspace" ?
        JR      NZ,PCH        20 07
        LD      A,CLEFT       3E 08         ;"cleft"
        CALL    SCREDI        CD 00 00      ;ausfuehren ..
        LD      (HL),#20      36 20         ;und Zeichen loeschen
PCH     CP      CH            FE 1C         ;"cursor home" ?
        JR      NZ,PCBL       20 05
        LD      HL,SBEG       21 BB BB      ;HL ← links oben
NCRS    JR      NEWCRS        18 15
PCBL    CP      CBL           FE 1D         ;"cbl" ?
        JR      NZ,EXIT       20 14
        INC     HL            23
        EX      HL,DE         EB            ;DE ← Cursor+1
        LD      HL,SBEG       21 BB BB      ;HL ← Anf. v. Zeile 1
        LD      BC,LLEN       01 AA 00      ;B ← 00 ; C ← LLEN
FINDLN  ADD     HL,BC         09            ;HL ← naechste Zeile
        PUSH    HL            E5
        SBC     HL,DE         ED 52         ;unterhalb v. Cursor ?
        POP     HL            E1
        JR      C,FINDLN      38 F9         ;C=1, wenn noch nicht
        SBC     HL,BC         ED 42         ;eine Zeile zurueck
NEWCRS  LD      (CURS),HL     22 CC CC      ;CURS ← neue Adresse
EXIT    POP     HL            E1            ;Register
        POP     DE            D1            ;zurueck
        POP     BC            C1            ;poppen
        POP     AF            F1
        RET                   C9            ;RUECKSPRUNG.......
```

```
;     Texteditor I/O-Routinen fuer "Terminal-Systeme" :
;     -----------------------------------------------------
;
SCREDI  EQU     #0000           ;SCREDIT ( siehe Bild 2 )
OUTCH   EQU     #AAAA           ;Systemroutine druckt Zeichen, dessen
                          ;ASCII-Code im Accu, auf dem Bildschirm aus.
CURS    EQU     #CCCC           ;Cusoradresse von SCREDIT
CR      EQU     #0D             ;ASCII fuer "carriage return"
CDOWN   EQU     #0A             ;ASCII fuer "line feed"
CUP     EQU     #0B             ;ASCII fuer "Cursor move up"
INCH    EQU     #1111           ;Tastaturroutine (s. unten und Text)
;
        ORG     #EEEE           ;beliebig
;
OUTPUT  CALL    OUTCH           CD AA AA       ;Zeichen zum Bildschirm
        CALL    SCREDI          CD 00 00       ;Z. in Buffer eintragen
        RET                     C9
;
; INPUT1 :    INPUT-Routine vom Typ 1      ( s. Text )
;
INPUT1  CALL    INCH            CD 11 11       ;Zeichen holen
        CP      CR              FE 0D          ;"carriage return" ?
        JR      Z,GETEND        28 05          ;Z=1, wenn ja
        CALL    SCREDI          CD 00 00       ;Z. in Buffer eintragen
        JR      INPUT1          18 F4          ;naechstes Zeichen holen
GETEND  LD      A,CDOWN         3E 0A          ;Cursor um eine Zeile
        CALL    SCREDI          CD 00 00       ;nach unten .
        LD      A,CUP           3E 0B          ;Cursor zurueck
        CALL    SCREDI          CD 00 00       ;==> Scroll kompensiert
        LD      DE,(CURS)       ED 5B CC CC    ;DE ← Zeilenende + 1
        LD      A,CR            3E 0D          ;"carriage return" auch
        JP      SCREDI          C3 00 00       ;im Buffer ausfuehren .
;
; INPUT2 :    INPUT-Routine vom Typ 2      ( s. Text )
;
INPUT2  CALL    INCH            CD 11 11       ;Tastaturabfrage
        AND     A               A7             ;Taste gedrueckt ?
        JR      Z,INPUT2        28 FA          ;Z=0, wenn ja
        CP      CR              FE 0D          ;"cr" gedrueckt ?
        JR      Z,GEND          28 05          ;Z=1, wenn ja
        CALL    OUTPUT          CD EE EE       ;Zeichen drucken
        JR      INPUT2          18 F1          ;naechstes Zeichen
GEND    LD      A,CDOWN         3E 0A          ;"line feed"
        CALL    OUTPUT          CD EE EE       ; .... u.s.w.
;weiter wie bei Typ1, jedoch OUTPUT anstatt SCREDI !!!
;
;
```

Bild 3. Für Computer, bei denen die Zentraleinheit die Ein/Ausgabe terminal-ähnlich vornimmt, benötigt der Texteditor diese Hilfsroutinen

her den Screen-Editor unverändert übernehmen können. Aus diesem Grunde arbeitet SCREDIT nicht mit einer Code-Tabelle, sondern nach dem „Strickleiterprinzip“. Diese Methode erlaubt auf einfachste Weise das Ändern sowie Hinzufügen eigener Steuerzeichen.

Die Implementierung des Z80-Texteditors sieht dann folgendermaßen aus: Die OUTPUT-Routine gibt das im Akku befindliche ASCII über die schon vorhandene Systemroutine an den Bildschirm aus und trägt dieses Zeichen via SCREDIT in den Textbuffer im RAM ein. Die INPUT-Routine dagegen ist sehr systemabhängig. Prinzipiell gibt es hier zwei Möglichkeiten:

1. Die INCH-Routine des Betriebssystems liest ein Zeichen von der Tastatur und druckt es anschließend auf dem Bildschirm aus.
2. Die INCH-Routine ist der TRS80-Systemroutine vergleichbar (s. oben).

Während erster Fall z. B. für den Elzet 80 zutrifft, ist der letztere jedoch häufiger, denn eine Routine, die die Tastaturschnittstelle bedient, dürfte in jedem Betriebssystem vorhanden sein. In *Bild 3* sind die für beide Typen erforderlichen Assemblerprogramme aufgelistet.

Ein Kapitel für sich: Implementierung auf Nascom-1

Der Nascom-1 ist ein gutes Beispiel dafür, wie sich ein System hartnäckig gegen jede Konvention durchsetzen kann. Das nicht gerade üppige Angebot an Steuerzeichen (CR=1F, FF=1E, Backstep=1D) kann man nur über Hexcodes auskosten, die derart unüblich sind, daß Nascom-1-Programme garantiert auf keinem anderen System laufen. Obwohl auch dieser Computer einen Videospeicher besitzt, wäre es aufgrund der begrenzten Möglichkeiten müßig zu versuchen, den Texteditor mittels TRS80-ähnlichen Routinen anzupassen. *Bild 4* zeigt deshalb das dokumentierte Hexlisting zweiter Unterprogramme, die die INCH- und OUTCH-Routinen des NASBUG-Monitors (dort heißen sie KBD und CRT) ersetzen können.

Die OUTCH-Routine gleicht im Prinzip dem in Bild 2 dargestellten Screen-Editor. Sie ist jedoch dem Bildschirmformat des Nascom angepaßt und unterstützt auch die für den Z80-Texteditor sehr nützliche Headerzeile durch spezielle Steuerzeichen. Die INCH-Routine „zaubert“ aus 47 Tasten 116 verschiedene ASCII-Codes, von denen 8 frei programmierbar sind. Abgesehen vom doch sehr begrenzten Bilschirmformat (nur 16 Zeilen zu je 48 Zeichen), bietet der Nascom

```
   *** komfortable I/O-Systemroutinen fuer NASCOM-1 ***
        ( ersetzen CRT und KBD im NASBUG-Monitor )

1.      K E Y T A B L E :
( Anfangsadresse =0F87 muss in 0C43+0C44 stehen,
  ausserdem muss 0C3F den Wert 5E enthalten .  )

0F87 * 08 88 09 14 9C 9B A3 92 C2 BA B2 AA A2 98 A0 29 *
0F97 * 0A 21 19 1A 1C 1B 23 12 42 3A 32 2A 22 18 20 A9 *
0FA7 * 8A A1 99 8D AC C1 93 BB B3 C3 90 C0 AD B8 B0 A8 *
0FB7 * B1 B9 A5 9D A4 95 B4 C5 B5 91 AB C4 BD BC 9A 0D *
0FC7 * 89 94 0B 0B 2C 41 13 3B 33 43 10 40 2D 38 30 28 *
0FD7 * 31 39 25 1D 24 15 34 45 35 11 2B 44 3D 3C *  *  *

2.  K B D - Routine  ( Tastaturabfrage )
    0C68   = ASCII der zuletzt gedrueckten Taste
    0C69   = Counter fuer Auto-Repeat
    0C6C   = ASCII an der Stelle des Cursors (0C18)
    0C6D   = Counter fuer blinkenden Cursor

0C6E * E5 D5 C5 CD[69 00]30 08 6F 26 00 22[68 0C]18 3E *
0C7E * 3E 02 CD[4A 00]06 07 3E 01 CD[4A 00]DB 00 2F E6 *
0C8E * 3F 20 16 10 F2 3E 01 CD[4A 00]DB 00 2F E6 3F 28 *
0C9E * 1D FE 10 28 19 FE 20 28 15 2A[68 0C]24 3E E0 BC *
0CAE * 28 06 AF 22[68 0C]18 06 7D 26 C3 37 18 F5 F5 3A *
0CBE *[6D 0C]3C FE 60 20 11 2A[18 0C]7E FE 7F 20 06 3A *
0CCE *[6C 0C]77 18 02 36 7F AF 32[6D 0C]F1 30 28 FE 1D *
0CDE * 20 02 3E 18 FE 1E 20 02 3E 08 FE 1F 20 02 3E 0D *
0CEE * FE 5B 20 02 3E 0B FE 5D 20 02 3E 0A FE 5E 20 02 *
0CFE * 3E 09 FE 5C 20 03 AF 18 2F 21[01 0C]CB 6E 2B 28 *
0D0E * 06 D6 60 30 02 C6 40 CB 6E 28 06 FE 61 38 02 D6 *
0D1E * 20 CB 56 28 12 FE 18 20 02 3E 1D FE 0C 20 02 3E *
0D2E * 1E FE 0D 20 02 3E 1F 37 C1 D1 E1 C9 *  *  *  *  *

Unterprogramme fuer CRT-Routine :

CRT0 = Video-RAM (0800-0BFF) loeschen & formatieren
0D3A * 0E 10 21[FA 07]06 10 36 00 23 10 F8 06 30 36 20 *
0D4A * 23 10 FB 0D 20 EF 36 00 C9 *  *  *  *  *  *  *  *
CRT1 = scroll up
0D53 * 21[40 08]11[00 08]01[7A 03]ED B0 EB 0E 01 18 DC *
CRT2 = scroll down
0D63 * 21[89 0B]11[C9 0B]01[8F 03]ED B8 18 EF *  *  *  *

3.    C R T - Routine  ( ASCII auf Bildschirm drucken )
      0C6A+0C6B = Beginn der aktuellen Bildschirmzeile

0D70 * E5 D5 C5 F5 FE 20 38 36 2A[18 0C]23 7E A7 20 18 *
0D80 * 11[B9 0B]EB ED 52 30 06 CD[53 0D]11[7A 08]21 10 *
0D90 * 00 19 22[6A 0C]1B 18 04 ED 5B[18 0C]7E 32[6C 0C]*
0DA0 * 22[18 0C]3E 5F 32[6D 0C]F1 12 C1 D1 E1 C9 FE 1D *
0DB0 * 20 02 3E 18 FE 1E 20 02 3E 0C FE 1F 20 02 3E 0D *
0DC0 * 2A[18 0C]FE 01 20 12 3A[6C 0C]77 21[CA 0B]22[18 *
0DD0 * 0C]22[6A 0C]7E 32[6C 0C]AF FE 02 20 19 11 10 00 *
0DE0 * 3E 20 77 32[6C 0C]23 7E A7 20 01 19 01[BA 0B]E5 *
0DF0 * ED 42 E1 38 EB AF FE 03 20 0E 3E 20 32[6C 0C]7E *
0E00 * A7 28 05 36 20 23 18 F7 FE 04 20 1D 54 5D 01 00 *
0E10 * 00 23 7E A7 28 12 23 0C 7E A7 20 FA 62 6B 23 7E *
0E20 * ED B0 32[6C 0C]EB 36 20 AF FE 05 20 21 01 00 00 *
0E30 * 3A[6C 0C]77 23 7E A7 28 0D 23 0C 7E A7 20 FA 2B *
0E40 * 54 5D 1B EB ED B8 3E 20 23 77 32[6C 0C]AF FE 06 *
0E50 * 20 04 CD[47 0C]AF FE 07 20 0D 21[00 0C]CB 6E 28 *
0E60 * 04 CB AE 18 02 CB EE FE 08 20 29 3A[6C 0C]77 2B *
0E70 * 7E A7 20 18 11 10 00 ED 52 E5 11[0A 08]E0 52 E1 *
0E80 * 38 12 E5 11 2F 00 ED 52 22[6A 0C]E1 7E 32[6C 0C]*
0E90 * 22[18 0C]AF FE 09 20 21 3A[6C 0C]77 23 7E A7 20 *
0EA0 * 10 11 10 00 19 E5 11[BA 0B]ED 52 E1 30 0B 22[6A *
0EB0 * 0C]7E 32[6C 0C]22[18 0C]AF FE 0A 20 2A 3A[6C 0C]*
0EC0 * 77 11 40 00 19 E5 01[BA 0B]ED 42 E1 30 10 7E 32 *
0ED0 *[6C 0C]22[18 0C]2A[6A 0C]19 22[6A 0C]18 08 CD[53 *
0EE0 * 0D]3E 20 32[6C 0C]AF FE 0B 20 16 3A[6C 0C]77 11 *
0EF0 * C0 FF 19 E5 01[09 08]ED 42 E1 30 D2 CD[63 0D]18 *
0F00 * E0 FE 0C 20 11 CD[3A 0D]21[0A 08]22[18 0C]22[6A *
0F10 * 0C]7E 32[6C 0C]AF FE 0D 20 07 3E 0A CD[70 0D]3E *
0F20 * 1A FE 0E 20 05 21[00 0C]CB D6 FE 0F 20 05 21[00 *
0F30 * 0C]CB 96 FE 18 20 0B 3E 20 77 32[6C 0C]3E 08 CD *
0F40 * 70 0D FE 19 20 06 3A[6C 0C]77 18 BC FE 1A 20 0F *
0F50 * 3A[6C 0C]77 2A[6A 0C]22[18 0C]7E 32[6C 0C]AF FE *
0F60 * 18 30 1A FE 10 38 16 DD 21[50 0C]11 03 00 E6 0F *
0F70 * 28 05 DD 19 3D 18 F7 F1 C1 D1 E1 DD E9 3E 5F 32 *
0F80 *[6D 0C]F1 C1 D1 E1 C9 *  *  *  *  *  *  *  *  *
```

```
S T E U E R Z E I C H E N :

Code  Funktion                                        Taste
-----------------------------------------------------------
 00   nul : Cursor helltasten                          ---
 01   Cursor zum Headeranfang                          @A
 02   ab Cursor Rest des Bildschirms loeschen          @B
 03   ab Cursor Rest der Bildschirmzeile loeschen      @C
 04   Zeichen loeschen, Rest der Zeile rueckt nach     @D
 05   Blank einfuegen, ....Zeile wird aufgeschoben     @E
 06   $NMI-Reflection aufrufen (==> 0C47)              @F
 07   GROSS/kleinschreibung umschalten (0C00:bit5)     @G
 08   Cursor nach links                            shift BS
 09   Cursor nach rechts                           shift SPACE
 0A   Cursor nach unten, line feed up              shift NL
 0B   Cursor nach oben , line feed down            shift 0
 0C   clear screen & cursor home                       @L
 0D   carriage return                                  NL
 0E   BS/@L/NL erzeugen NAS-Codes 1D/1E/1F (0C00:      @N
 0F   BS/@L/NL erzeugen ASC-Codes 18/0C/0D  bit2)      @O
 10   function 0 : Sprung nach 0C50                    @0
 11   function 1 : Sprung nach 0C53                    @1
 12   function 2 : Sprung nach 0C56                    @2
 13   function 3 : Sprung nach 0C59                    @3
 14   function 4 : Sprung nach 0C5C                    @4
 15   function 5 : Sprung nach 0C5F                    @5
 16   function 6 : Sprung nach 0C62                    @6
 17   function 7 : Sprung nach 0C65                    @7
 18   Backspace                                        BS
 19   Cursor home                                      @Y
 1A   Cursor zurueck zum Zeilenanfang                  @Z
 1D   entspricht 18
 1E   entspricht 0C
 1F   entspricht 0D
-----------------------------------------------------------
' @ ' hat die Funktion einer CONTROL-Taste
Das Zeichen '@' kann man durch 'shift @' erzeugen.
Werden die functions nicht benoetigt, dann sollten in
0C50,0C53.....,0C65 jeweils ein RET-Befehl C9 stehen.
Haelt man eine Taste laenger als ca. 1/2 Sekunde gedrueckt,
dann wird sie automatisch mit 15 Zeichen/sec wiederholt.


I/O-Routinen zur Anpassung an Z80-Texteditor :
==============================================

INPUT:    CD 6E 0C 30 FB FE 0D 28 05 CD 70 0D 18 F2
          3E 0A CD 70 0D 3E 0B CD 70 0D ED 5B 18 0C
          3E 0D C3 70 0D
OUTPUT:   C3 70 0D

fuer NASSYS :
INPUT:    CF FE 0D 28 03 F7 18 F8 3E 14 F7 3E 13 F7
          ED 5B 29 0C 3E 0D F7 C9
OUTPUT:   F7 C9
```

Bild 4. Der Computer Nascom-1 verwendet ziemlich unübliche ASCII-Steuerzeichen. Dieses Hilfsprogramm macht ihn kompatibel zum Texteditor

mit Hilfe dieser beiden Maschinenprogramme sämtliche für die Textverarbeitung notwendigen Bildschirm-Editiermöglichkeiten:
- blinkender, nicht zerstörender und frei beweglicher Cursor,
- Groß- und Kleinschreibung
- automatische Dauerfunktion (Autorepeat),
- scroll down (über „Cursor up“ am oberen Bildrand),
- gängige Steuerzeichen, ASCII-kompatibel.

Die Anpassung des Texteditors geschieht mit diesen beiden Routinen wie beim TRS80.

Der Vollständigkeit halber sei noch erwähnt, daß es für den Nascom inzwischen ein neues und wesentlich verbessertes Betriebssystem NASSYS gibt, das dem TRS80 vergleichbare „on-screen-editing“-Möglichkeiten bietet. Bild 4 enthält deshalb außerdem Hinweise zur Anpassung an NASSYS.

Herwig Feichtinger

Kreuzkorrelation per Programm

Im letzten Heft haben wir bereits eine sehr effiziente Methode zur Erkennung gestörter, verrauschter Signale im Tonfrequenzbereich kennengelernt: die Autokorrelation. Hier folgt nun ein noch leistungsfähigeres Verfahren zur Tonerkennung per Software, das auf der Kreuzkorrelation beruht.

Während die Autokorrelation [1] das digitalisierte Eingangssignal zeitverschoben mit sich selbst vergleicht, wird bei der sogenannten Kreuzkorrelation ein empfangsseitig erzeugtes Signal bzw. Bitmuster als Referenz verwendet. Dies hat den Vorteil, daß das Referenzsignal nicht schon ebenso gestört ist wie das Eingangssignal. Deshalb lassen sich auch stark verrauschte Signale mit der Kreuzkorrelation noch erkennen, bei denen die Autokorrelation längst versagt. Außerdem fällt die Kreuzkorrelation nicht auf Harmonische, also ganzzahlige Vielfache der Eingangsfrequenz herein.

Die Phasenlage interessiert nicht

Um die Übereinstimmung des Referenzsignals mit dem Eingangssignal prüfen zu können, müßte die Phasenlage des Eingangssignals bekannt sein, was normalerweise nicht der Fall ist. Während bei einer reinen Hardware-Kreuzkorrelation zur Vermeidung dieses Problems meist ein sin/cos-Referenzsignal nebst zwei Multiplizierern eingesetzt wird [2], verschiebt die hier vorgestellt Software-Lösung das 8-Bit-Referenzmuster so lange, bis eine optimale Übereinstimmung mit dem digitalisierten Eingangssignal festgestellt wird. Im Beispiel von *Bild 1* ist dazu ein zweimaliges Rotieren des Referenzmusters, das stets genau eine Periode des erwarteten Signals darstellt, erforderlich. Der Referenz-Grundwert hex F0 entspricht dem Bitmuster 11110000, was eine digitalisierte Sinus-Periode bzw. ein symmetrisches Rechtecksignal darstellt.

Parameter leicht änderbar

Das in *Bild 2* als Assembler-Listing abgedruckte 6502-Programm ist für die Adressenbelegung der Computer AIM-65 bzw. PC-100 ausgelegt und verwendet wie [1] deren Kassetten-Port als Eingang, so daß keinerlei zusätzliche Hardware erforderlich ist.

Das Programm zählt einfach innerhalb eines gewissen Zeitintervalls, dessen Dauer in Zelle 0000 einstellbar ist, die Anzahl der mit dem Referenzmuster übereinstimmenden Bits. Diese erreicht natürlich bei der Sollfrequenz, die durch den Inhalt von Zelle 0001 festgelegt ist, ein Maximum. Das Referenz-Bitmuster, normalerweise hex F0, ist in Zelle 0002 zu schreiben, und der Inhalt der Zelle 0003 bestimmt schließlich, wieviele Bytes nacheinander mit dem digitalisierten Eingangssignal aufgefüllt werden, bevor der Test auf Übereinstimmung mit dem Referenzmuster erfolgt. Dies bestimmt im wesentlichen die Bandbreite, hat aber leider wegen der Beeinflussung der Programmlaufzeit auch Einfluß auf die Sollfrequenz. Die Frequenz ergibt sich zu:

$$f = \frac{10^6}{8 \cdot (19 + 11\ \mathrm{LEN} + 5\ \mathrm{FREQ})}\ \mathrm{Hz}$$

Die am Schluß des Listings in Bild 2 angegebene Bytefolge stellt die Parameter auf die Erkennung der bei Funkübertragungen üblichen Ruffrequenz von 1750 Hz ein. Je nach LEN (0003) benötigt das Programm eine entsprechende Zahl von Bytes ab Adresse 000A als Software-Schieberegister für das Eingangssignal.

Programmkern voll relokatibel

Der „Kern" des 6502-Programms, der von 0C00 bis 0C62 reicht, ist in seiner Adressenlage frei verschiebbar. Systemspezifisch ist lediglich die Adresse des Eingangsports (hier A800). Als Ergebnis steht die Zahl aller innerhalb der Meßdauer als gleich erkannter Bits hexadezimal vierstellig in ERGH und ERGL zur Verfügung.

Diese Zahl wird vom letzten Programmabschnitt, der wegen der Adressen von NUMA und CRLF AIM-65/PC-100-spezifisch ist, auf das Display ausgegeben, was hier nur zu Demonstrationszwecken dient. Als Kriterium für das Vorhandensein eines Tones mit der Sollfrequenz kann beim angegebenen Beispiel das Überschreiten eines Wertes von etwa hex 0300 gewertet werden.

Die Empfindlichkeit läßt sich wie bei der Autokorrelation durch Verlängern der Meßdauer in 0000 steigern. Es ist kein Problem, noch Signale zu erkennen, die die gleiche Amplitude wie das störende Rausch-Signal besitzen – ein Beweis, daß die digitale Signalverarbeitung der analogen keineswegs unterlegen sein muß.

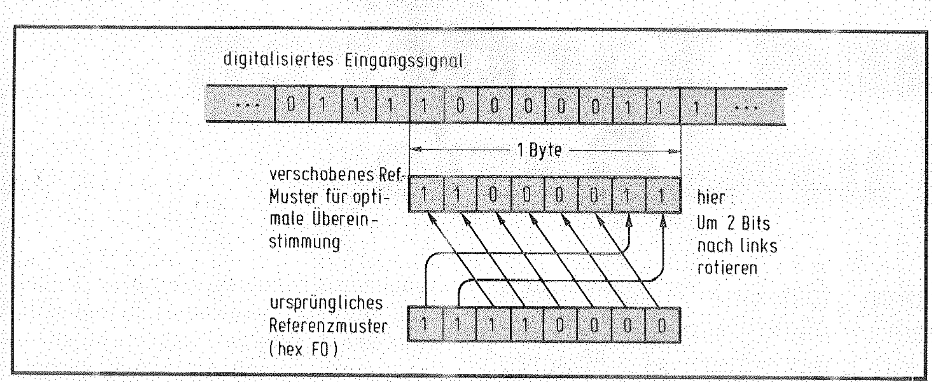


Bild 1. Elimination der Phaseninformation durch Verschieben des Referenz-Bitmusters

Literatur

[1] Tonerkennung per Software. mc 1981, Heft 4.
[2] Kreuzkorrelator zur Messung von Nf-Spektren. ELEKTRONIK 1981, Heft 12.

```
0000          ;KORRELATION MIT
0000          ;REFERENZ-BITMUSTER
0000                 *=0
0000          ;BENUTZERVARIABLE
0000 PER             *=*+1      ;MESSDAUER
0001 FREQ            *=*+1      ;FREQUENZ
0002 REFP            *=*+1      ;REFERENZ
0003 LEN             *=*+1      ;BYTEZAHL
0004          ;PROGRAMMVARIABLE
0004 SUM             *=*+1
0005 CNT             *=*+1      ;ZAEHLER
0006 CNT1            *=*+1
0007 MAX             *=*+1      ;MAX.SUM
0008 ERGL            *=*+1      ;ERGEBNIS
0009 ERGH            *=*+1
000A SMPL            *=$10C     ;SAMPLES
010C
010C          ;AIM-65-ADRESSEN
010C          4C000C JMP CCOR   ;TASTE F1
010F CRLF            =$E9F0
010F NUMA            =$EA46     ;DISPLAY
010F PB              =$A800     ;PB7=INPUT
010F                 *=$C00
0C00
0C00          ;PROGRAMMSTART
0C00 CCOR     A900   LDA #0     ;ALLES
0C02          8508   STA ERGL   ;LOESCHEN
0C04          8509   STA ERGH
0C06          A500   LDA PER    ;MESS-
0C08          8506   STA CNT1   ;DAUER
0C0A          ;BITMUSTER LESEN
0C0A REC      A503   LDA LEN
0C0C          0A     ASL A      ;LEN X 8
0C0D          0A     ASL A      ;ERGIBT
0C0E          0A     ASL A      ;BIT-
0C0F          8505   STA CNT    ;ANZAHL
0C11 REC1     A603   LDX LEN
0C13          AD00A8 LDA PB
0C16          2A     ROL A
0C17 REC2     3609   ROL SMPL-1,X
0C19          CA     DEX
0C1A          D0FB   BNE REC2
0C1C          A401   LDY FREQ   ;VER-
0C1E REC3     88     DEY        ;ZOEGERUNG
0C1F          D0FD   BNE REC3
0C21          C605   DEC CNT
0C23          D0EC   BNE REC1
0C25          ;MUSTERVERGLEICH
0C25          A908   LDA #8     ;REF. 8MAL
0C27          8505   STA CNT    ;SCHIEBEN
0C29          4A     LSR A      ;MAX=4
0C2A          8507   STA MAX
0C2C COM0     A502   LDA REFP   ;REF.
0C2E          2A     ROL A      ;ROTIEREN
0C2F          2602   ROL REFP
0C31          A900   LDA #0
0C33          8504   STA SUM    ;SAMPLES
0C35          A603   LDX LEN    ;MIT
0C37 COM1     A008   LDY #8     ;REFER.
0C39          B509   LDA SMPL-1,X   ;VERGL.
0C3B          4502   EOR REFP
0C3D COM2     2A     ROL A      ;GLEICHE
0C3E          B002   BCS COM3   ;BITS
0C40          E604   INC SUM    ;ZAEHLEN
0C42 COM3     88     DEY
0C43          D0F8   BNE COM2
0C45          CA     DEX
0C46          D0EF   BNE COM1
0C48          A504   LDA SUM    ;MAXIMAL-
0C4A          C507   CMP MAX    ;WERT
0C4C          9002   BCC COM4   ;SPEICHERN
0C4E          8507   STA MAX
0C50 COM4     C605   DEC CNT
0C52          D0D8   BNE COM0
0C54          A507   LDA MAX    ;MAX.-WERT
0C56          18     CLC        ;ZUM ER-
0C57          6508   ADC ERGL   ;GEBNIS
0C59          8508   STA ERGL   ;ADDIEREN
0C5B          9002   BCC NOCY
0C5D          E609   INC ERGH
0C5F NOCY     C606   DEC CNT1
0C61          D0A7   BNE REC
0C63          ;AUSGABE AUF AIM-65
0C63          20F0E9 JSR CRLF   ;DISPLAY
0C66          A509   LDA ERGH   ;LOESCHEN
0C68          2046EA JSR NUMA
0C6B          A508   LDA ERGL   ;ERGEBNIS
0C6D          2046EA JSR NUMA   ;ANZEIGEN
0C70          4C000C JMP CCOR
0C73          ;DATEN FUER 1750 HZ
0C73          ;0000: 20 02 F0 04
0C73                 .END
```

Bild 2. Assemblerlisting des AIM-65/PC-100-Demonstrationsprogramms

---

Jürgen Plate

Suchen und Sortieren in Pascal und Basic

5. Teil

Unser Literaturverzeichnis-Programm stellt eine Zusammenfassung des bisher Gelernten dar. Seine Funktionsweise und Bedienung wurde bereits im letzten Heft besprochen; als Abschluß folgen nun noch die kompletten Programmlistings.

Gleichzeitig ist unsere kleine Serie über Such- und Sortierverfahren in den zwei wichtigsten höheren Tischcomputer-Programmiersprachen beendet; wir hoffen, es hat Ihnen ein wenig Spaß gemacht.

```
  1 PROGRAM LISTE(INPUT/,OUTPUT,DATA);
  2 (*
  3    LITERATURVERZEICHNIS FUER ZEITSCHRIFTENARTIKEL.
  4
  5    DIE DATEI ' DATA' ENTHAELT DIE LITERATURREFERENZEN,
  6    DIE AUSGABE ERFOLGT NACH OUTPUT.
  7    WEITERE BESCHREIBUNG SIEHE NUNKSCHAU-ARTIKEL
  8
  9    ANMERKUNG: DER '/' HINTER INPUT IST NICHT STD-PASCAL
 10
 11 *)
 12
 13 CONST N = 499;  (*LAENGE DER HASH-TABLE *)
 14       LENGTH = 1000;  (* LAENGE DER DATENLISTE *)
 15       BLANK30 = '                              '; (* 30 BLANKS *)
 16       KEYL = 30    (* KEYLAENGE *);
 17
 18 TYPE  KEYWORD = PACKED ARRAY [1..KEYL] OF CHAR ; (* STRING *)
 19       LISTARRAY = ARRAY [1..LENGTH,1..3] OF INTEGER;
 20               (* LITERATURLISTE : HEFT,SEITE,NAECHSTER *)
 21       TABLE = ARRAY [0..N] OF RECORD
 22                                  KEY : KEYWORD;
 23                                  START : 0..LENGTH
 24                                END;
 25
 26 VAR   LIST : LISTARRAY;
 27       TAB  : TABLE;
 28       DATAPOINTER : 0..LENGTH (* ZEIGT AUF DAS NAECHSTE FREIE
 29                                  LISTENELEMENT *);
 30       KOMMANDO : CHAR;
 31       DATA : TEXT;  (* TYPE TEXT = FILE OF CHAR *)
 32
 33
 34 PROCEDURE GENERATENEWLIST;
 35    VAR I : 1..LENGTH;
 36        J : 0..N;
 37    BEGIN
 38    FOR I := 1 TO LENGTH DO
 39       FOR J := 1 TO 3 DO LIST[I,J] := 0;
 40    FOR J := 0 TO N DO
 41       BEGIN
 42       TAB[J].KEY := BLANK30;
 43       TAB[J].START := 1;
 44       END;
 45    END;
 46
 47
 48
 49 PROCEDURE PUTDATA;
 50    VAR I : 1.. LENGTH;
 51        J : 0..N;
 52    BEGIN
 53    REWRITE(DATA);
 54    FOR J := 0 TO N DO
 55       WRITELN(DATA,TAB[J].KEY,' ',TAB[J].START :1);
 56    FOR I := 1 TO DATAPOINTER-1 DO
 57       WRITELN(DATA,LIST[I,1]:1,' ',LIST[I,2]:1,' ',
 58                    LIST[I,3]:1);
 59    END;
 60
 61
 62
 63 PROCEDURE GETDATA;
 64    VAR I : 1..LENGTH;
 65        J : 0..N;
 66        K : 1..KEYL;
 67    BEGIN
 68    RESET(DATA);
 69    FOR J := 0 TO N DO
 70       BEGIN
 71       (* STRINGS MUESSEN ZEICHENWEISE GELESEN WERDEN *)
 72       FOR K := 1 TO KEYL DO READ(DATA,TAB[J].KEY[K]);
 73       READ(DATA,TAB[J].START);
 74       READLN(DATA);
 75       END;
 76    DATAPOINTER := 1;
 77    WHILE NOT EOF(DATA) DO
 78       BEGIN
 79       READ(DATA,LIST[DATAPOINTER,1],LIST[DATAPOINTER,2],
 80                 LIST[DATAPOINTER,3]);
 81       READLN(DATA);
 82       DATAPOINTER := DATAPOINTER + 1;
 83       END;
 84    END;
 85
 86
 87
 88 PROCEDURE GETLINE(VAR K : KEYWORD; VAR H,S : INTEGER);
 89    VAR I : 0..KEYL;
 90        CH :CHAR;
 91    BEGIN
 92    READLN   (* MASCHINENABHAENGIG *);
 93    K := BLANK30;
 94    H := 0; S := 0;
 95    READ(INPUT,K[1]);
 96    IF K[1]<>'$' THEN
 97       BEGIN
 98       I := 1; READ(CH);
 99       WHILE (CH<>',') AND NOT EOLN DO
100          BEGIN
101          IF I< KEYL THEN BEGIN I := I+1; K[I] := CH END;
102          READ(CH);
103          END;
104       IF NOT EOLN THEN READ(H,CH);
105       IF NOT EOLN THEN READ (S);
106       END;
107    END;
108
109
110 FUNCTION HASH(X :KEYWORD) : INTEGER;
111    VAR I : 1..5;
112        H : INTEGER;
113    BEGIN
114    H:=0;
115    FOR I := 1 TO 5 DO H:= H*10 + ORD(X[I]) ;
116    HASH := H MOD N;
117    END;
118
119
120
121 PROCEDURE INSERT(KEY : KEYWORD; HEFT,SEITE : INTEGER);
122    VAR GEFUNDEN : BOOLEAN;
123        MARKE,INDEX : 0..N;
124        ZEIGER :  0..LENGTH;
```

```
125    BEGIN
126    GEFUNDEN := FALSE;
127    INDEX := HASH(KEY); MARKE := INDEX;
128    REPEAT
129       (* SUCHE KEY IN TABELLE *)
130       IF TAB[INDEX].KEY = BLANK30 THEN
131          BEGIN (* NEUEINTRAG *)
132          GEFUNDEN := TRUE;
133          TAB[INDEX].KEY := KEY;
134          TAB[INDEX].START := DATAPOINTER;
135          LIST[DATAPOINTER,1] := HEFT;
136          LIST[DATAPOINTER,2] := SEITE;
137          DATAPOINTER := DATAPOINTER + 1;
138          END
139       ELSE
140          IF TAB[INDEX].KEY = KEY THEN
141             BEGIN(* GETROFFEN *)
142             GEFUNDEN := TRUE;
143             ZEIGER := TAB[INDEX].START;
144             WHILE LIST[ZEIGER,3]<>0 DO
145                ZEIGER := LIST[ZEIGER,3];
146             LIST [ZEIGER,3] := DATAPOINTER;
147             LIST[DATAPOINTER,1] := HEFT;
148             LIST[DATAPOINTER,2] := SEITE;
149             DATAPOINTER := DATAPOINTER + 1;
150             END
151          ELSE
152             BEGIN (* WEITERSUCHEN *)
153             IF INDEX = N THEN INDEX := 0 ELSE INDEX :=
                                                  INDEX + 1;
154             IF INDEX = MARKE THEN
155                BEGIN WRITELN(' --- TABLE OVERFLOW ---');
156                      GEFUNDEN := TRUE END;
157             END;
158    UNTIL GEFUNDEN;
159    IF DATAPOINTER = LENGTH THEN WRITELN(' --- LIST
                                          OVERFLOW ---');
160    END;
161
162
163
164 PROCEDURE ENTER;
165    VAR KEY : KEYWORD;
166        HEFT,SEITE : INTEGER;
167    BEGIN
168    WRITELN(  ' PLEASE ENTER: KEY,ISSUE,PAGE');
169    WRITELN(' FINISH WITH $ ');
170    REPEAT
171       GETLINE(KEY,HEFT,SEITE);
172       IF( KEY[1]<>'$') AND (KEY<>BLANK30) THEN INSERT(KEY,HEFT
                                                   ,SEITE);
173       WRITELN(' O.K.');
174    UNTIL KEY[1]='$';
175    END;
176
177
178
179 PROCEDURE SEARCH(KEY : KEYWORD);
180    VAR INDEX,MARKE : 0..N;
181        ZEIGER,K : 0..LENGTH;
182        GEFUNDEN : BOOLEAN;
183
```

```
184    BEGIN
185    GEFUNDEN := FALSE; K := 0;
186    INDEX := HASH(KEY); MARKE := INDEX;
187    REPEAT
188       IF TAB[INDEX].KEY = KEY THEN
189          BEGIN  (* GEFUNDEN *)
190          GEFUNDEN := TRUE;
191          WRITELN(' ':3,KEY);
192          WRITE(' ':5);
193          ZEIGER := TAB[INDEX].START;
194          REPEAT
195             WRITE(' ':3,LIST[ZEIGER,1]:3,'/',LIST[ZEIGER,2]:1);
196             K := K + 1; IF K MOD 5 = 0 THEN
197                BEGIN
198                WRITELN; WRITE(' ':5);
199                END;
200             ZEIGER := LIST[ZEIGER,3];
201          UNTIL ZEIGER = 0;
202          IF K MOD 5<>5 THEN WRITELN;
203          WRITELN;
204          WRITELN;
205          END
206       ELSE
207          BEGIN
208          IF INDEX = N THEN INDEX := 0 ELSE INDEX := INDEX + 1;
209          IF INDEX = MARKE THEN
210            BEGIN GEFUNDEN := TRUE;
211            WRITELN(' --- ',KEY, ' NOT FOUND ---');
212            END;
213          END;
214    UNTIL GEFUNDEN;
215    END;
216
217
218
219 PROCEDURE QUESTION;
220    VAR KEY : KEYWORD;
221          I : 0..KEYL;
222    BEGIN
223    WRITELN(' PLEASE ENTER KEYWORD');
224    READLN;
225    I := 0;
226    KEY := BLANK30;
227    WHILE NOT EOLN AND (I<KEYL) DO
228       BEGIN I:=I+1; READ(KEY[I]) END;
229    SEARCH(KEY);
230    END;
231
232
233
234 PROCEDURE LISTTABLE;
235    VAR KEYS : ARRAY[0..N] OF KEYWORD;
236          KEY : KEYWORD;
237          I,J,K : INTEGER;
238          ENDE : BOOLEAN;
239    BEGIN
240    K := 0;
241    FOR I := 0 TO N DO
242       IF TAB[I].KEY <> BLANK30 THEN
243          BEGIN KEYS[K] := TAB[I].KEY; K := K + 1 END;
244    K := K - 1;
245    FOR I := 1 TO K DO
```

Seite 65

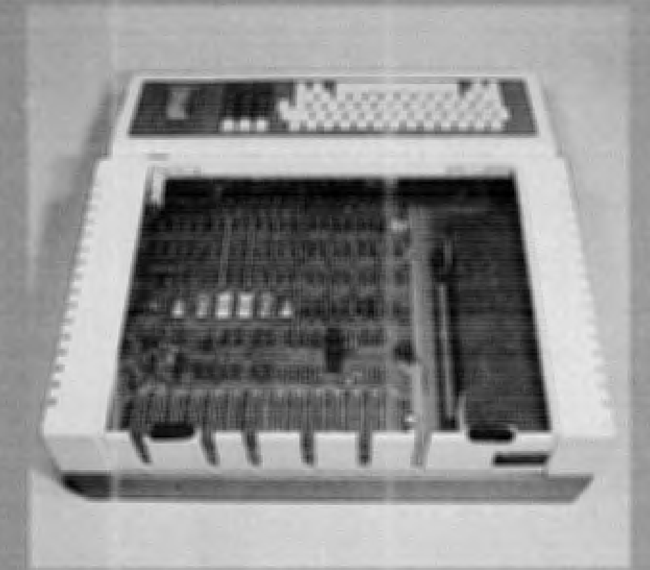
Wir stellen vor:

pearcom®

Mikrocomputer mit PAL-COLOR

Leistungsdaten:

- Großes Softwareangebot verfügbar
- 24 I/O Bits in der Grundausführung
- 40 × 24 Zeichen in Groß- und Kleinschrift
- Option 80 × 24 Zeichen
- 48K RAM auf 96K RAM erweiterbar
- 6502 CPU
- Buskompatibel mit Z80 Microsoftkarte
- Dadurch CP/M bzw. Microsoftbasic
- Option Cobol, Pascal, Fortran etc.
- 16 Farben FBAS-Video und HF Ausgang
- PAL Standard auf dem Motherboard
- [illegible]
- [illegible]
- Tastatur mit 8 Funktionstasten und 10er Block
  - Zeichensatz mit 256 Zeichen im EPROM
  - RS 232, Centronics parallel, IEC, V24 etc. als Interface verfügbar
  - 16K Drucker Speicher Interface



Verkauf über den Fachhandel



pearcom®

Distribution in Deutschland:
KLEINOFEN COMPUTER, Kölner Straße 49, 4000 Düsseldorf, Tel. 369191 – Tlx 8582848

6809-Befehlsübersicht

Der Prozessor 6809 gehört zu den leistungsfähigsten 8-Bit-CPUs, die heute auf dem Markt sind, und verfügt intern über eine 16-Bit-Struktur. Dr. H. J. Neukirchen stellte die 6809-Operationscodes in einer übersichtlichen, kompakten Tabelle zusammen.

Op.Symbol	Op.Code der Adressierungsarten				CC-Flags	Bemerkungen
Akku A/B	IMMEDIATE ( ) Byte	DIRECT 2 Byte	EXTENDED 3 Byte	POSTBYTE 2-5 Byte	N Z V C	1) D ≙ [A | B]
	Arithmetisch-logische Befehle					M ≙ Memoryzelle
ADD_	8B/CB (2)	9B/DB	BB/FB	AB/EB	X X X X	A/B ← A/B + M
ADDD	C3 (3)	D3	F3	E3	X X X X	D ← D + M 2)
ADC_	89/C9 (2)	99/D9	B9/F9	A9/E9	X X X X	A/B ← A/B + M + C
SUB_	80/C0 (2)	90/D0	B0/F0	A0/E0	X X X X	A/B ← A/B − M
SUBD	83 (3)	93	B3	A3	X X X X	D ← D − M 2)
SBC_	82/C2 (2)	92/D2	B2/F2	A2/E2	X X X X	A/B ← A/B − M − $\bar{C}$
CMP_	81/C1 (2)	91/D1	B1/F1	A1/E1	X X X X	A/B − M
CMPD	1083 (4)	1093	10B3	10A3	X X X X	D − M 2)
CMPX	8C (3)	9C	BC	AC	X X X X	X − M 2)
CMPY	108C (4)	109C	10BC	10AC	X X X X	Y − M 2)
CMPS	118C (4)	119C	11BC	11AC	X X X X	S − M 2)
CMPU	1183 (4)	1193	11B3	11A3	X X X X	U − M 2)
BIT_	85/C5 (2)	95/D5	B5/F5	A5/E5	X X X X	A/B ∧ M
AND_	84/C4 (2)	94/D4	B4/F4	A4/E4	X X X X	A/B ← A/B ∧ M
OR_	8A/CA (2)	9A/DA	BA/FA	AA/EA	X X X X	A/B ← A/B ∨ M
EOR_	88/C8 (2)	98/D8	B8/F8	A8/E8	X X X X	A/B ← A/B ⊻ M
MUL	3D (1 Byte) vorz.lose Multiplik				− X − X	D ← A × B 3)
	Daten-Transport Befehle					
LD_	86/C6 (2)	96/D6	B6/F6	A6/E6	X X 0 −	A/B ← M
ST_	−	97/D7	B7/F7	A7/E7	X X 0 −	M ← A/B
LDD	CC (3)	DC	FC	EC	X X 0 −	D ← M 2)
STD	−	DD	FD	ED	X X 0 −	M ← D 2)
LDX	8E (3)	9E	BE	AE	X X 0 −	X ← M 2)
STX	−	9F	BF	AF	X X 0 −	M ← X 2)
LDY	108E (4)	109E	10BE	10AE	X X 0 −	Y ← M 2)
STY	−	109F	10BF	10AF	X X 0 −	M ← Y 2)
LDS	10CE (4)	10DE	10FE	10EE	X X 0 −	S ← M 2)
STS	−	10DF	10FF	10EF	X X 0 −	M ← S 2)
LDU	CE (3)	DE	FE	EE	X X 0 −	U ← M 2)
STU	−	DF	FF	EF	X X 0 −	M ← U 2)
TFR	1F (2)	Op., d s			− − − −	R_d ← R_s
EXG	1E (2)				− − − −	R_d ↔ R_s
	Schiebebefehle					M bzw. A/B
	A / B 4)					
ASL	48/58	08	78	68	X X − X	
ASR	47/57	07	77	67	X X − X	
LSR	44/54	04	74	64	X X − X	
ROL	49/59	09	79	69	X X − X	
ROR	46/56	06	76	66	X X − X	
	Inkrementieren, Dekrementieren					
INC	4C/5C	0C	7C	6C	X X X −	M ← M + 1
DEC	4A/5A	0A	7A	6A	X X X −	M ← M − 1
	Komplementieren, Negieren, Löschen, Test					
COM	43/53	03	73	63	X X 0 1	1-Komplement $\bar{M}$
NEG	40/50	00	70	60	X X X X	2-Komplement $\bar{M}$+1
CLR	4F/5F	0F	7F	6F	0 1 0 0	Alle Bit in M = 0
TST	4D/5D	0D	7D	6D	X X 0 −	M − 0

d/s − Digit	0	1	2	3	4	5	8	9	A	B
Register	D	X	Y	U	S	PC	A	B	CC	DP

1) H-Flag wird nur durch ADD, ADC signifikant verändert (für evtl. DAA)
2) M bedeutet einen 2-Byte Operanden, niedere Adresse angeben
3) C = 1 ≙ B_7 = 1 (nützlich für Runden)
4) 1-Byte Befehl; in der Spalte "Bemerkungen" ist M durch A/B zu ersetzen

Op.-Symbol	Op.Code der Adressierungsarten					CC-Flags	Bemerkungen
	IMPLIED nur Op.	IMMED. ( ) Byte	DIRECT 2 Byte	EXTENDED 3 Byte	POSTBYTE 2-5 Byte	N Z V C	
	Ändern des X, Y, S, U - Registers						
LEAX	Ein 5- und 8-Bit-Offset wird für die Addition als komeg. Zahl auf 16 Bit erweitert				30	- X - -	X ← eff. Adr.
LEAY					31	- X - -	Y ← eff. Adr.
LEAS					32	- - - -	S ← eff. Adr.
LEAU					33	- - - -	U ← eff. Adr.
ABX	3A	Add. von B als vorz.lose Zahl				- - - -	X ← X + B
	Ändern der Flags im CC-Register [E F H I N Z V C]						
ANDCC	-	1C (2)	-	-	-	X X X X	CC ← CC ∧ M
ORCC	-	1A (2)	-	-	-	X X X X	CC ← CC ∨ M

Bedingte und unbedingte Sprünge

Kurz-Offset		Lang-Offset		Sprungbedgg.
BPL	2A	LBPL	102A	N = 0
BMI	2B	LBMI	102B	N = 1
BNE	26	LBNE	1026	Z = 0
BEQ	27	LBEQ	1027	Z = 1
BVC	28	LBVC	1028	V = 0
BVS	29	LBVS	1029	V = 1
BCC	24	LBCC	1024	C = 0
BCS	25	LBCS	1025	C = 1
BGE	2C	LBGE	102C	N⊻V = 0
BLT	2D	LBLT	102D	N⊻V = 1
BGT	2E	LBGT	102E	Cv(N⊻V) = 0
BLE	2F	LBLE	102F	Cv(N⊻V) = 1
BHI	22	LBHI	1022	CvZ = 0
BLS	23	LBLS	1023	CvZ = 1
BRA	20	LBRA	16	(3 Byte)

relative Adressierung bei Branch-Befehlen:

Kurz-Offset → [Op. |]

Lang-Offset → [Op. Code | |]

Offset als komegative Zahl

Bei erfüllter Sprungbedingung:

PC ← PC + n + Offset

sonst: PC ← PC + n (n = Befehlslänge (in Byte))

Op.-Symbol	IMPLIED	IMMED.	DIRECT	EXTENDED	POSTBYTE	Bemerkungen
JMP	-	-	0E	7E	6E	unbedingter Sprung PC ← eff. Adr.

Unterprogramm- und Stack-Technik

Op.-Symbol	IMPLIED	IMMED.	DIRECT	EXTENDED	POSTBYTE	Bemerkungen
BSR	8D	LBSR 17 (3 Byte)	(relat. Adress.)			Stack ← Rückkehradr., PC ← eff. Adr.
JSR	-	-	9D	BD	AD	Stack ← Rückkehradr., PC ← eff. Adr.
RTS	39	-	-	-	-	PC ← Stack (Rückkehradr.)
PSHS	-	34 (2)				Stack ← maskierte Reg.
PULS	-	35 (2)				maskierte Reg. ← Stack
PSHU	-	36 (2)				Stack ← maskierte Reg.
PULU	-	37 (2)				maskierte Reg. ← Stack

Bedeutung der Maskenbit:

PSHS/PULS: [PC U Y X DP B A CC]

Bit 7 ——— 0

PSHU/PULU: [PC S Y X DP B A CC]

hohe Adr. ——→ niedrige Adr. im Stack

Verschiedene Operationen

Op.-Symbol	Op.Code		N Z V C	Bemerkungen
SEX	1D	* Sign Extend *	X X - -	Komeg. Erweit. von B auf D
DAA	19	BCD-Korrektur in A	X X X	Nur hinter ADDA, ADCA; V-Flag undefiniert
NOP	12	* No Operation *		

Interrupt-Technik E F I - Flag 5) Reg.folge im Stack s. PSHS

Op.-Symbol	Op.Code		E F I	Bemerkungen
CWAI	-	3C (2)	1 X X	CC ← CC ∧ M; Stack ← alle Reg.; Warten (6)
SYNC	13		- - -	* Synchronisation mit Interrupt * (6)
		Int.-Vektor:		
SWI	3F	FFFA, FFFB	1 1 1	"Software-Interrupt", Stack ← alle Register; fallende Priorität
SWI2	103F	FFF4, FFF5	1 - -	
SWI3	113F	FFF2, FFF3	1 - -	
RTI	3B	CC ← Stack; dann bei E=0: PC ← Stack, bei E=1: alle Reg. ← Stack		

Hardware-Interrupt:

		Int.-Vektor	E F I	fallende Priorität: maskierbar	
RS	Pin 37	FFFE, FFFF	- 1 1	nein	"Reset"
NMI	Pin 2	FFFC, FFFD	1 1 1	nein	Stack ← alle Reg.
FIRQ	Pin 4	FFF6, FFF7	0 1 1	durch E-Flag	Stack ← PC, CC
IRQ	Pin 3	FFF8, FFF9	1 - 1	durch I-Flag	Stack ← alle Reg.

5) Interrupt setzt F-bzw. I-Flag erst, nachdem das CC-Reg. gestackt ist.
6) Ein maskierter Interrupt hat bei CWAI keine Wirkung, bei SYNC wird das Programm mit dem nachfolgenden Befehl fortgesetzt.

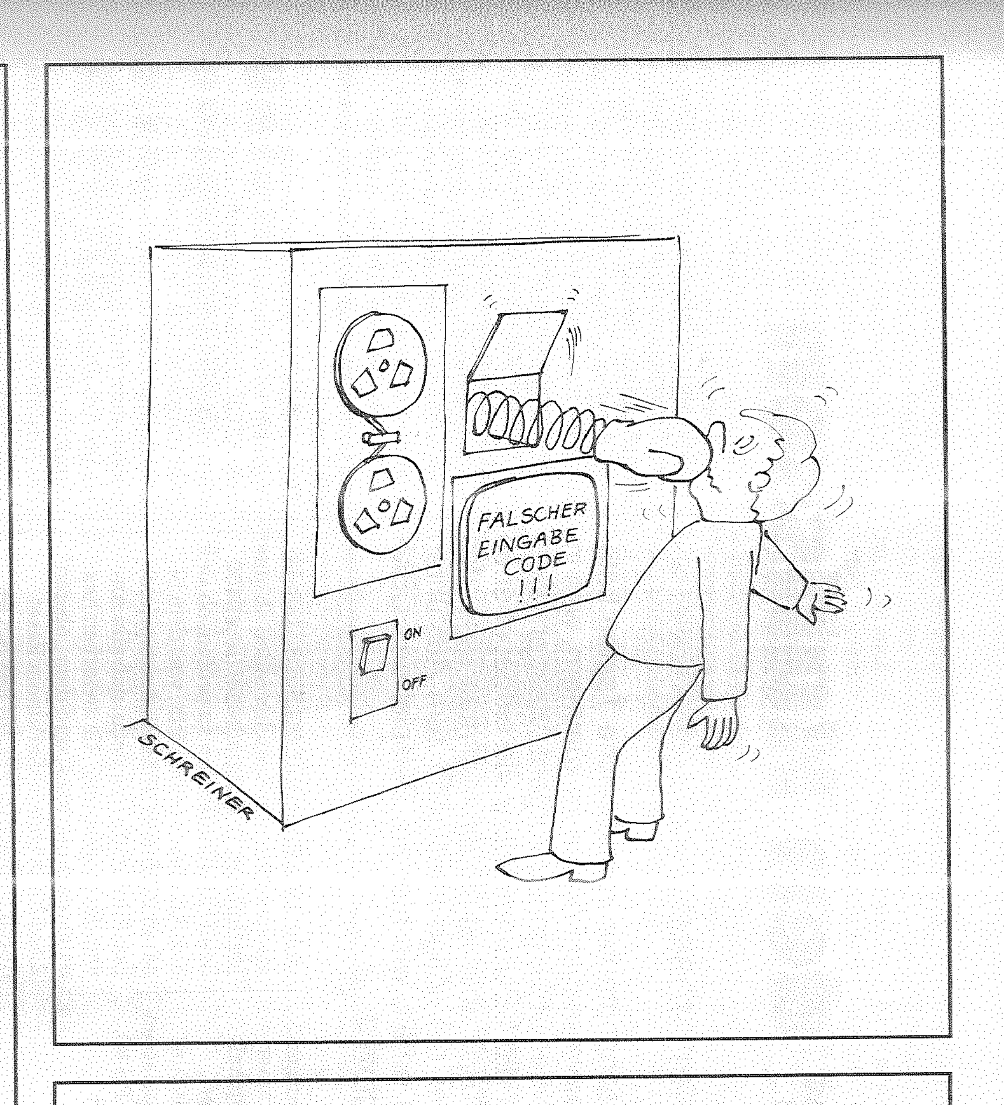
```
246       BEGIN
247       ENDE := FALSE;
248       KEY := KEYS[I];
249       J := I-1;
250       WHILE NOT ENDE AND (J>=0) DO
251          IF KEY<KEYS[J] THEN
252             BEGIN KEYS[J+1] := KEYS[J]; J := J-1 END
253          ELSE ENDE := TRUE;
254       KEYS[J+1] := KEY;
255       END;
256    WRITELN(' HOW MANY COPIES DO YOU WANT');
257    READLN; READ(I);
258    REPEAT
259       WRITELN(' ':25,'LITERATURSTELLEN');
260       WRITELN(' ':25,'================');
261       WRITELN; WRITELN; WRITELN;
262       FOR J := 0 TO K DO
263         SEARCH(KEYS[J]);
264       WRITELN; WRITELN; WRITELN;
265       I := I - 1
266    UNTIL I = 0;
267    END;
268
269
270
271
272 (* MAIN *)
273 BEGIN
274 DATAPOINTER :=1;
275 WRITELN(' ENTER COMMAND');
276 READLN; READ(KOMMANDO);
277 GENERATENEWLIST;
278 IF KOMMANDO <> 'N' THEN GETDATA;
279 REPEAT
280    IF KOMMANDO IN ['E', 'Q','P','N'] THEN
281       CASE KOMMANDO OF
282       'E' : ENTER;
283       'Q' : QUESTION;
284       'P' :LISTTABLE;
285       'N' : GENERATENEWLIST;
286       END
287 ELSE
288    WRITELN(' --- ILLEGAL COMMAND,REENTER ---');
289 WRITELN(' ENTER COMMAND');
290 READLN; READ(KOMMANDO);
291 UNTIL KOMMANDO = 'S';
292 WRITELN(' GOOD BYE');
293 PUTDATA;
294 END.
```

```
00005 REM * *************************
00010 REM * LITERATURSTELLEN-PROGRAMM
00020 REM * *************************
00030 REM * VARIABLEN:
00040 REM *   L(L1,3) LITERATURSTELLEN (HEFT,SEITE,NACHFOLGER)
00050 REM *   HASH-TABELLE: K$ FUER DIE SCHLUESSEL
00060 REM *                 S FUR DIE STARTINDICES IN L
00070 REM *                 K$(N),S(N)
00080 REM *   F$ NAME DES DATENFILES
00090 REM *   D ZEIGT AUF DAS ERSTE FREIE ELEMENT VON L
00100 REM * KONSTANTE FUER DIE FELDLAENGEN BESETZEN
00110 N = 499
00120 L1 = 1000
00125 DIM L(1000,3),K$(499),S(499),K9$(499)
00130 REM * * *  BEGINN HAUPTPROGRAMM  * * *
00140 D=1
00150 PRINT " PLEASE ENTER NAME OF DATAFILE";
00160 INPUT F$
00170 GOSUB 1000
00180 REM * GENERIEREN LEERER LISTE
00190 PRINT " ENTER COMMAND"
00200 INPUT A$
00210 IF A$="N" THEN 240
00220 GOSUB 2000
00230 REM * EINLESEN DES DATENFILES
00240 REM * KOMMANDO BEARBEITEN
00250 IF A$ = "E" THEN 340
00260 IF A$ = "Q" THEN 360
00270 IF A$ = "P" THEN 380
00280 IF A$ = "N" THEN 400
00290 IF A$ = "S" THEN 420
00300 PRINT " --- ILLEGAL COMMAND ---"
00310 PRINT " --- ALLOWED ARE  E,Q,N,P  OR  S ---"
00320 GOTO 440
00330 REM * AUFRUFE DER FUNKTIONEN
00340 GOSUB 3000
00350 GOTO 440
00360 GOSUB 4000
00370 GOTO 440
00380 GOSUB 5000
00390 GOTO 440
00400 GOSUB 1000
00410 GOTO 440
00420 PRINT " GOOD BYE"
00425 GOSUB 6000
00430 STOP
00440 REM * EINLESEN NAECHSTES KOMMANDO
00450 PRINT " ENTER COMMAND"
00460 INPUT A$
00470 GOTO 240
00480 REM * ENDE HAUPTPROGRAMM
00490 REM **************************************************
00500 REM **************************************************
00510 REM *
00515 REM * DEFINIEREN DER HASH- FUNKTION
00520 REM *
00530 DEF FNH (X$)
00540   REM  SUBSTR IST IDENTISCH MIT MIDSTR
00550 I = 0
00560 FOR J = 1 TO 5
00570    L9 = 0
00580    FOR K = 1 TO 63
```

```
00590       IF SUBSTR(X$,J,1)<>CHR$(K) THEN L9 = K
00600       NEXT K
00610    I = I*10 + L9
00620 NEXT J
00630 FNH = I - INT(I/N)*N
00640 FNEND
00650 REM *
00660 REM *
00670 REM *
00680 REM *
01000 REM ****************************************************
01001 REM *  GENERIEREN EINER LEEREN LISTE
01002 REM ****************************************************
01010 FOR I = 1 TO L1
01020    FOR J = 1 TO 3
01030       L(I,J) = 0
01040    NEXT J
01050 NEXT I
01060 FOR I = 1 TO N
01070    K$(I) = " "
01080    S(I) = 1
01090 NEXT I
01110 RETURN
02000 REM ****************************************************
02001 REM *  LESEN DES DATENFILES
02002 REM ****************************************************
02010 FILE#1 = F$
02025 REM * FILE IST GLEICHBEDEUTEND MIT OPEN
02030 RESTORE #1
02035 REM * DATEI AUF ANFANG STELLEN
02040 FOR J = 1 TO N
02050    INPUT#1,K$(J),S(J)
02060 NEXT J
02070 D = 1
02080 IF END#1 THEN 2120
02090 INPUT#1,L(D,1),L(D,2),L(D,3)
02100 D = D + 1
02110 GOTO 2080
02120 CLOSE#1
02130 RETURN
03000 REM ****************************************************
03001 REM *  EINGABE NEUER DATEN VON DER TASTATUR
03002 REM ****************************************************
03010 PRINT " PLEASE ENTER  KEY,ISSUE,PAGE"
03020 PRINT " FINISH WITH  $,0,0"
03030 INPUT K1$,H1,S1
03040 IF K1$ = "$" THEN 3080
03050 GOSUB 7000
03060 PRINT "O. K."
03070 GOTO 3030
03080 RETURN
04000 REM ****************************************************
04001 REM *  FRAGE NACH EINTRAEGEN ZU EINEM KEYWORD
04002 REM ****************************************************
04010 PRINT "PLEASE ENTER KEYWORD"
04020 INPUT K1$
04030 GOSUB 8000
04040 RETURN
05000 REM ****************************************************
05001 REM *  AUFLISTEN DER GANZEN DATEN
05002 REM ****************************************************
05020 K=1
```

```
05030 FOR I = 1 TO N
05040    IF K$(I) =" " THEN 5070
05050    K9$(K) = K$(I)
05060    K = K + 1
05070    NEXT I
05080 K = K - 1
05090 FOR I = 2 TO K
05100    H$ = K9$(I)
05110    J = I - 1
05120    IF H$>=K9$(J) THEN 5160
05130    K9$(J+1) = K9$(J)
05140    J = J - 1
05150    IF J<>0 THEN 5120
05160    K9$(J+1) = H$
05170 NEXT I
05175 K8 = K
05180 PRINT"HOW MANY COPIES DO YOU WANT"
05190 INPUT I9
05200 PRINT TAB(25),"LITERATURSTELLEN"
05210 PRINT TAB(25),"================"
05220 PRINT
05230 PRINT
05240 PRINT
05250 FOR J9 = 1 TO K8
05260    K1$ = K9$(J9)
05270    GOSUB 8000
05280 NEXT J9
05290 I9 = I9 - 1
05300 IF I9>0 THEN 5200
05310 RETURN
06000 REM ****************************************************
06001 REM *  ZURUECKSCHREIBEN DER DATEN AUF DAS DATENFILE
06002 REM ****************************************************
06010 FILE#1 = F$
06020 RESTORE#1
06030 FOR J = 1 TO N
06040    PRINT#1,K$(J);",";S(J);","
06050 NEXT J
06060 FOR J = 1 TO D-1
06070    PRINT#1,L(J,1);",";L(J,2);",";L(J,3);","
06080 NEXT J
06090 CLOSE#1
06100 RETURN
07000 REM ****************************************************
07001 REM *  EINFUEGEN EINES EINTRAGS IN DIE LISTE/TABELLE
07002 REM ****************************************************
07003 REM *
07004 REM * PARAMETER:
07005 REM *    K1$  EINZUFUEGENDER SCHLUESSEL
07006 REM *    H1      HEFTNUMMER
07007 REM *    S1      SEITENNUMMER
07008 REM *
07010 G = 0
07020 I = FNH(K1$)
07030 M = I
07040 REM SUCHE NACH DEM SCHLUESSEL
07050 IF K$(I)<>" " THEN 7130
07060 G = 1
07070 K$(I) = K1$
07080 S(I) = D
07090 L(D,1) = H1
07100 L(D,2) = S1
```

```
07110 D = D + 1
07120 GOTO 7310
07130 IF K$(I)<>K1$ THEN 7240
07140 G = 1
07150 Z = S(I)
07160 IF L(Z,3) = 0 THEN 7190
07170 Z = L(Z,3)
07180 GOTO 7160
07190 L(Z,3) = D
07200 L(D,1) = H1
07210 L(D,2) = S1
07220 D = D + 1
07230 GOTO 7310
07240 IF I = N THEN 7270
07250 I = I + 1
07260 GOTO 7280
07270 I = 1
07280 IF I <> M THEN 7310
07290 PRINT " --- TABLE OVERFLOW ---"
07300 G = 1
07310 IF G = 0 THEN 7040
07320 IF D<L1 THEN 7340
07330 PRINT " --- LIST OVERFLOW ---"
07340 RETURN
08000 REM ******************************************************
08001 REM *  SUCHEN IN DER TABELLE NACH EINEM SCHLUESSEL
08002 REM ******************************************************
08003 REM *
08004 REM * PARAMETER:
08005 REM *    K1$  ZU SUCHENDER SCHLUESSEL
08006 REM *
08010 G = 0
08020 I = FNH(K1$)
08030 K = 0
08040 M = I
08050 REM BEGINN DER SUCHE
08060 IF K$(I)<>K1$ THEN 8220
08070 G = 1
08080 PRINT"    ";K1$
08090 Z = S(I)
08100 PRINT "    ";L(Z,1);"/";L(Z,2);
08110 K = K + 1
08120 IF K<5 THEN 8150
08130 PRINT
08140 K = 0
08150 Z = L(Z,3)
08160 IF Z<>0 THEN 8100
08170 IF K = 0 THEN 8190
08180 PRINT
08190 PRINT
08200 PRINT
08210 GOTO 8290
08220 IF I = N THEN 8250
08230 I = I + 1
08240 GOTO 8260
08250 I = 1
08260 IF I<>M THEN 8290
08270 PRINT " --- ";K1$;" NOT FOUND ---"
08280 G = 1
08290 IF G = 0 THEN 8050
08300 RETURN
09000 END
```



Zitat des Monats

„Das Produkt Computer ist nicht gleichzusetzen mit Coca Cola oder Hamburgern von McDonald."

CHIP 12/1981, S. 17

Herwig Feichtinger

Hinter den Kulissen

Wie Basic-Interpreter arbeiten

Wer an seinem Basic-Tischcomputer irgendwelche Tasten betätigt, braucht sich normalerweise nicht darum zu kümmern, was im „Hirn" des Computers, in der CPU, genau vorgeht. Vielmehr sieht es so aus, als verstünde die CPU unmittelbar die eingegebenen Basic-Befehle. In Wirklichkeit aber ist sie ein dummes Ding, das erst durch den Interpreter die nötige Intelligenz zum Ausführen komplexer Befehlszeilen erhält, wie hier am Beispiel des CBM-Basic-Interpreters gezeigt wird.

Die meisten Basic-Interpreter heutiger Tischcomputer wurden von der amerikanischen Firma Microsoft entwickelt und ähneln sich daher sehr in der Grundstruktur. Die Aussagen, die hier zum CBM-Basic gemacht werden, lassen sich daher prinzipiell auch für TRS-80, AIM-65, PC-100 oder Apple II anwenden.

Nach dem Einschalten

Schalten wir einen Computer ein, so sorgt eine kleine Logikschaltung – ein Monoflop – dafür, daß den Bruchteil einer Sekunde lang an einem bestimmten Anschluß der CPU, des Mikroprozessors, ein Impuls anliegt, der einen Reset auslöst. Dieser Reset tut nichts weiter als dafür zu sorgen, daß die CPU sich von einer ganz bestimmten, hardwaremäßig im Prozessor festgelegten Adresse im ROM zwei Bytes holt, diese wiederum als Adresse interpretiert und dort mit der Ausführung eines (ebenfalls noch im ROM) befindlichen Maschinenprogramms beginnt. Beim CBM sieht das so aus: Die CPU 6502 holt sich die Reset-Adresse stets aus den (dezimalen) Speicherzellen 65532 und 65533 (entspr. hex FFFC und FFFD). Diese beiden Bytes ergeben zusammen die 16-Bit-Zahl 64721, und genau an dieser Adresse beginnt die CPU mit der Programmausführung.
Zunächst werden der Stackpointer (Stapelzeiger) sowie mehrere RAM-Speicherzellen initialisiert, die solche Dinge wie Interrupt-Vektoren und Betriebsparameter enthalten. Bei manchen Computern – so auch beim CBM – wird nach dem Reset außerdem überprüft, wieviel RAM als Arbeitsspeicher zur Verfügung steht. Dies führt die CPU durch, indem sie versucht, alle Speicherzellen ab einer bestimmten Anfangsadresse (hex 0400 beim CBM) daraufhin zu prüfen, ob sich hex AA einschreiben läßt, was dem binären Bitmuster 10101010 entspricht. Das führt natürlich dazu, daß alle Anwenderprogramme, die zwischen hex 0400 (dezimal 1024) und dem „oberen" RAM-Ende stehen, bei einem Reset durch Überschreiben mit hex AA gelöscht werden.
Selbstverständlich löscht das Maschinenprogramm, das die CPU nach einem Reset ausführt, auch den Bildschirm und schreibt solche Dinge wie „Commodore Basic" und „XXXX Bytes free" auf den Schirm. Dann aber läuft das Programm weiter zum sogenannten „Editor".

Programmeingabe per Editor

Der Editor ist ebenfalls ein in Maschinensprache geschriebener Programmteil im ROM des Computers und ermöglicht es dem Benutzer, Programme einzugeben oder zu ändern. Der CBM verfügt über einen bildschirmorientierten Editor, d. h. es ist möglich, mit Cursor-Steuerbewegungen einzelne Zeichen auf dem Bildschirm auszubessern und die so geänderten Zeilen komplett in den Arbeitsspeicher zu übernehmen; andere Computer besitzen besondere Editier-Befehle wie z. B. CHANGE, EDIT oder FETCH.
Der Editor sorgt dafür, daß eine eingetippte Programmzeile mit einer vorangestellten Zeilennummer nach dem Drücken der Return-Taste, die bekanntlich jede Eingabe abschließt, in den Arbeitsspeicher übernommen wird. Das geschieht sinnvollerweise aber nicht Zeichen für Zeichen, sondern wesentlich speicherplatzsparender: Für jeden Befehl, der dem Computer bekannt ist (also für jedes reservierte Wort), wird nur ein Byte abgespeichert. Daß es sich dabei um einen Kurzbefehl, ein „Token" handelt, ist daran zu erkennen, daß das höchstwertige Bit in diesem Byte immer 1 ist, während es bei allen unverändert übernommenen ASCII-Zeichen (außer bei Grafik-Zeichen, die von Ausführungszeichen eingeschlossen sind) stets Null ist.
Die vor jeder Programmzeile stehende Nummer wird ebenfalls nicht in Form von ASCII-Zeichen abgespeichert, sondern vom Editor in eine 2-Byte-Binärzahl umgewandelt. So braucht auch eine fünfstellige Zeilennummer wie z. B. 24563 nicht fünf, sondern nur zwei Bytes im Arbeitsspeicher.
Damit der Basic-Interpreter später bei der Ausführung des Benutzerprogramms weiß, wo er jeweils die nächste Zeilennummer findet, ohne lange danach suchen zu müssen, speichert der Editor zusätzlich am Anfang jeder Basic-Zeile noch eine 16-Bit-Adresse ab, nämlich die Startadresse der nächsten Befehlszeile.

Den Rest macht der Interpreter

Der Editor hat dem Interpreter nun schon eine Menge Arbeit abgenommen: Er hat herausgefunden, wo in den Zeilen zulässige Befehle stehen, hat sie zu einem Byte komprimiert und auch die Zeilennummern in ihr binäres Äquivalent umgewandelt.
Gibt der Benutzer nun eine Zeile ein, die keine vorangestellte Zeilennummer enthält, so verläßt die CPU das Editorprogramm und springt zum eigentlichen Interpreterprogramm. Nehmen wir an, der Benutzer hätte kühn RUN eingetippt und die Return-Taste gedrückt: Die Interpretation des Basic-Programms als eine Folge zahlloser Maschinenbefehle, die alle im ROM des Interpreters stehen, kann beginnen.
Die Hauptaufgabe des Interpreters ist es also zunächst einmal, jedem Token, jedem Basic-Befehlswort also, diejenige Adresse zuzuordnen, unter der er das

zugehörige Maschinenprogramm im ROM findet, z. B. für SIN. Dazu bedient er sich einer Tabelle, die ebenfalls im ROM steht: Der komprimierte 1-Byte-Basicbefehl dient als Index in diese Tabelle. Beim CBM steht die Tabelle ab der dezimalen Adresse 49152, und die Adresse des SIN-Maschinenprogramms findet die CPU in den beiden Bytes bei 49244 und 49245. Die 16-Bit-Zahl dort ergibt die Adresse 57311, und dort springt der Interpreter hin, um den Sinus-Wert des nachfolgenden Ausdrucks zu berechnen.
Das Microsoft-CBM-Basic komprimiert nur die Basic-Befehle; andere Basic-Interpreter, z. B. das DAI-8080-Basic, wandeln auch Dezimalzahlen in arithmetischen Ausdrücken schon während der Editierphase in Binärzahlen um.
Der CBM muß diese Dezimal-Binärumwandlung während des Programmlaufs tun, was ihn natürlich mehr Zeit kostet. Andere Interpreter, etwa der des HP-85, arbeiten grundsätzlich im BCD-Code, also dezimal, und können daher auf diese Umwandlung ganz verzichten – allerdings unter Inkaufnahme eines größeren Speicherplatzbedarfs für mehrstellige Zahlen.

Der Interpreter erkennt Fehler

Manche Computer, z. B. ZX-80 und ABC-80, erkennen syntaktische Fehler bei der Programmeingabe schon mit dem Editor, d. h. bevor das Programm mit RUN gestartet wird. Der CBM hingegen nimmt die Prüfung auf richtige Syntax erst nach RUN vor. Stünde etwa hinter SIN keine Klammer, so würde sich der Interpreter darüber mit „Syntax Error" beschweren.
Bei all diesem Komfort heutiger Basic-Interpreter sollte man aber nicht vergessen: Erkannt werden stets nur syntaktische Fehler, d. h. Fehler in der Schreibweise von Befehlen, und grobe Fehler in der Programmlogik, etwa der Versuch, ein Variablen-Array zweimal mit DIM zu dimensionieren. Echte logische Fehler merkt der Computer leider nicht – und manchmal auch nicht der Benutzer...

Unterprogramme ohne RETURN

Zuweilen steht man vor dem Problem, aus einem Unterprogramm an eine bestimmte Zeile des Basic-Hauptprogramms direkt zurückspringen zu müssen, z. B. wegen eines Eingabefehlers. Tut man dies statt RETURN mit einem GOTO-Befehl, so stimmt der Stapelzeiger (Stackpointer) nicht mehr. Bei einem folgenden Return-Befehl wird falsch zurückgesprungen. Der Rücksprung in höhere Schachtelebenen ist also nicht mehr möglich, falls eine mit GOSUB angesprungene Unterroutine mit einem GOTO verlassen wird.
In Maschinensprache läßt sich das Problem auf einfache Weise umgehen, indem man z. B. mit zweimaligem PLA den Stackpointer wieder korrigiert. Die meisten Basic-Interpreter sehen diese Möglichkeit aber leider nicht vor. Eine klare Maschinensprache-Erweiterung schafft hier Abhilfe.

Die Wirkung des Maschinenprogrammes

Hinter dem Befehl SYS(826) findet man exakt die gleichen Verhältnisse vor (FOR-NEXT-, GOSUB-Hierarchie) als hätte das Programm den das zugehörige GOSUB beendenden RETURN-Befehl absolviert.

```
              *
              * PLA PLA FUER BASIC
              * P. HEUMOS  DG2MAZ
              *

              INDEX EPZ 71       $47
              * 2001: 153        $99
              * 8001: 71         $47

              SUCHE EQU 49834    $C2AA
              * 2001: 49836      $C2AC
              * 8001: 45858      $B322

              TITEL EQU 23202    $5AA2
              * 2001: 24482      $5FA2
              * 8001: 23202      $5AA2

              ERROR EQU 50007    $C357
              * 2001: 50009      $C359
              * 8001: 46031      $B3FC

              INTER EQU 50884    $C6C4
              * 2001: 50869      $C6B5
              * 8001: 46922      $B74A

033A: A9FF     LDA #255          ; INITIALISIERT FUER ROUTINE
033C: 8547     STA INDEX
033E: 20AAC2   JSR SUCHE         ; SUCHT ERKENNUNGSBYTE AUF STAPEL
0341: 9A       TXS
0342: C98D     CMP #141          ; ERKENNUNGSBYTE FUER RETURN ?
0344: F008     BEQ RE1
0346: A216     LDX #22           ; NEIN: LADE ZEIGER FUER MELDUNG
0348: 2CA25A   BIT TITEL         ;  'RETURN WITHOUT GOSUB ERROR'
034B: 4C57C3   JMP ERROR         ; DRUCKE MELDUNG, READY

034E: A207    RE1 LDX #7         ; LETZTES RETURN VOM STAPEL NEHMEN
0350: 68      RE2 PLA
0351: CA       DEX
0352: D0FC     BNE RE2
0354: 4CC4C6   JMP INTER         ; SPRUNG IN DIE INTERPRETERSCHLEIFE


 INDEX       $47
 SUCHE       $C2AA
 TITEL       $5AA2
 ERROR       $C357
 INTER       $C6C4
 RE1         $034E
 RE2         $0350
```

Bild 1. Maschinenprogramm zur Stack-Korrektur für das Betriebssystem CBM 3001

```
W:033A-0341   A9 FF 85 99 20 AC C2 9A
W:0342-0349   C9 8D F0 08 A2 16 2C A2
W:034A-0351   5F 4C 59 C3 A2 07 68 CA
W:0352-0359   D0 FC 4C B5 C6 00 00 00
```

Bild 2. So sieht der Hex-Dump des PET-2001-Programms aus; es unterscheidet sich geringfügig von der 3001-Version

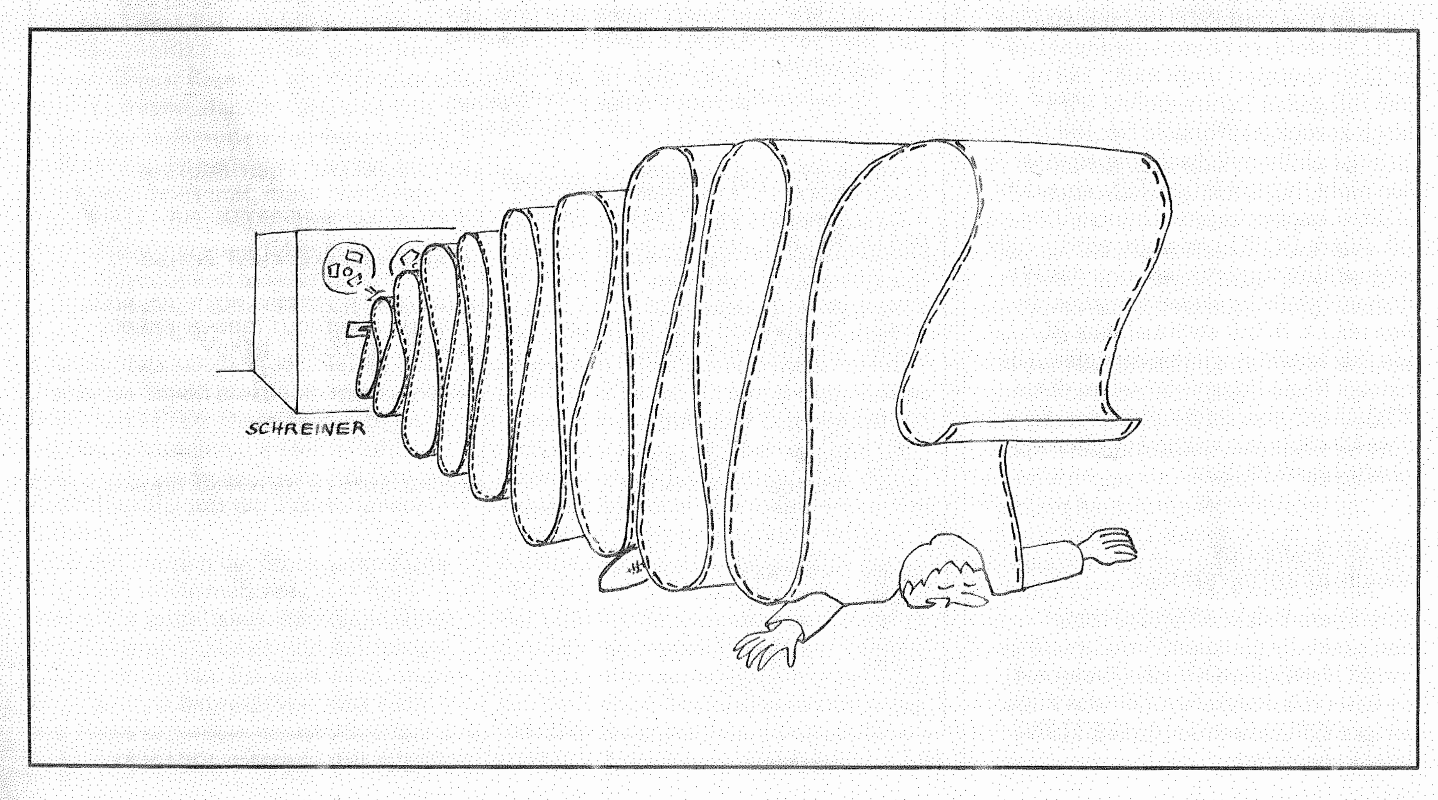
```
W:033A-0341   A9 FF 85 47 20 22 B3 9A
W:0342-0349   C9 8D F0 08 A2 16 2C A2
W:034A-0351   5A 4C FC B3 A2 07 68 CA
W:0352-0359   D0 FC 4C 4A B7 00 00 00
```

Bild 3. Version für das Betriebssystem CBM 4001/8001

```
100 STOP
110 TESTPROGRAMME STARTEN MIT :
120 RUN170
130 RUN200
140 RUN230
150 RUN280
160 :
170 GOSUB180
180 A=A+1:PRINT"A="A:GOTO170
190 :
200 GOSUB210
210 SYS826:A=A+1:PRINT"A="A:GOTO200
220 :
230 GOSUB240: PRINTA:NEXTA:END
240 FORA=0TO10
250 IFA<5THEN PRINTA:NEXTA
260 RETURN
270 :
280 GOSUB290: PRINTA:NEXTA:END
290 FORA=0TO10
300 IFA<5THEN PRINTA:NEXTA
310 SYS826:   PRINTA:NEXTA:END
READY.
```

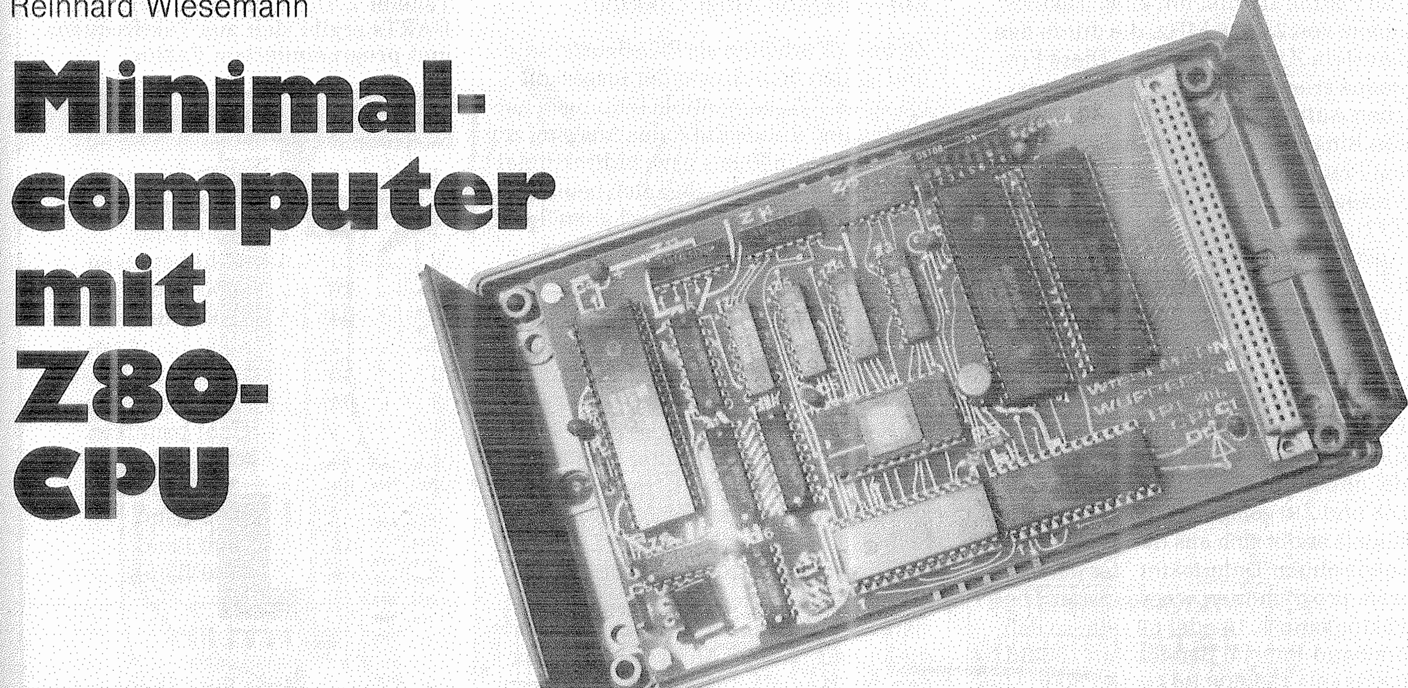
Bild 4. Die vier Basic-Teilprogramme demonstrieren die hilfreiche Wirkung der Stack-Korrektur mit SYS 826

In *Bild 1* ist die Erweiterung für Computer mit dem Betriebssystem CBM 3001 disassembliert aufgelistet. *Bild 2* enthält als Hex-Dump die entsprechende Version für den PET 2001, *Bild 3* für CBM 4001 und CBM 8001. Die Unterschiede ergeben sich aus unterschiedlichen Adressenbelegungen. Zu beachten ist, daß bestimmte Basic-Befehle des CBM 4001/8001 den Kassettenbuffer verwenden, in dem das Maschinensprache-Programm steht; sollten sich Schwierigkeiten ergeben, muß man das Programm in einen anderen Speicherbereich, z. B. geschützt ans RAM-Ende legen, wobei keine Veränderungen erforderlich sind. *Bild 4* zeigt schließlich ein Basic-Programm, das einige Testbeispiele enthält. RUN 170 demonstriert den Stack-Überlauf beim Rücksprung mit GOTO, RUN 230 bei falsch abgeschlossenen FOR-NEXT-Schleifen. RUN 200 und RUN 280 zeigen, wie der Einsatz der Maschinensprache-Erweiterung das Problem löst. *Paul Heumos*



Reinhard Wiesemann

Minimal-computer mit Z80-CPU



Nachdem wir in Heft 2/81 einen Minimalcomputer mit der CPU 6504 vorgestellt haben, folgt jetzt ein ähnliches System mit dem Z80. Gegenüber seinem Vorgänger bietet es mehr Ein-/Ausgabe-Leitungen und einen größeren RAM-Bereich. Auch die Programmspeicherkapazität ist höher. Geeignete Systeme für die Programmentwicklung sind Tischcomputer mit der CPU Z80 – z. B. Nascom, TRS-80 und Video-Genie.

In kleinen Steuerungen, Schnittstellen und ähnlichen Anwendungen lassen sich Mikrocomputer aufgrund ihrer hohen Leistungsfähigkeit und Flexibilität mit Vorteil einsetzen. Für diese Bereiche werden bereits zahlreiche Systeme angeboten, die im Baukastenprinzip an das spezielle Problem angepaßt werden können: Aus einem meist umfangreichen Angebot an CPU-, Speicher- und Ein-/Ausgabe-Platinen stellt man sich das gewünschte System zusammen. Der Vorteil dieses Prinzips liegt in der leichten und praktisch unbegrenzten Erweiterbarkeit eines so aufgebauten Mikrocomputers.

Nachteilig ist, daß kleine Systeme nahezu nicht realisierbar sind: Ein minimales System benötigt meist bereits drei Computerplatinen (CPU, Speicher, Ein-/Ausgabe), eine Buskarte sowie ein Netzteil mit meist mehreren Versorgungsspannungen. Berücksichtigt man dazu das dann häufig notwendige 19-Zoll-Gehäuse, so kann das so aufgebaute System in vielen Anwendungen schon allein aus preislichen Gründen nicht eingesetzt werden.

Der hier beschriebene Minimal-Mikrocomputer vermeidet diese Nachteile, da er speziell für den Einsatz in kleinen Systemen entwickelt wurde: Der gesamte Computerteil (inklusive CPU, RAM, EPROM, Ein-/Ausgabe) befindet sich auf einer einzigen Platine, und es ist nur eine 5-V-Versorgungsspannung nötig. Auf diese Weise können kleine Mikrocomputer sehr kompakt und preiswert in Standardgehäuse eingebaut werden. Ein vollständiges System benötigt neben der Computerplatine nur eine zusätzliche Karte, die die anwendungsspezifische Schaltung (Relais, Treiber usw.) und das Netzteil beinhaltet. *Bild 1* zeigt die Schaltung des Einplatinencomputers MMC-1. In den *Bildern 2* und *3* sind Platinenlayout und Bestückungsplan dargestellt. Die Platine ist unbestückt oder fertig bestückt vom Autor beziehbar (Postfach 20 16 05, 5600 Wuppertal 2).

Schaltungsdetails

Beim Einschalten der Versorgungsspannung wird die CPU durch das aus R8 und C7 bestehende RC-Glied zurückgesetzt. Der $\overline{\text{INT}}$-Eingang kann von einer speziellen Anwenderschaltung frei benutzt oder über C10 mit dem Baud-Rate-Generator verbunden werden. In diesem Fall erhält die CPU alle 256 Taktzyklen ein INT-Signal. Da auf der Platine keine Interrupt-Vektoren erzeugt werden, muß Interrupt-Mode 1 (Restart bei 038H) programmiert werden.

Die Karte bietet die Möglichkeit, bis zu 2 KByte EPROM (1 × Typ 2716 – Z10) sowie 2 KByte RAM (4 × Typ 2114 – Z2...Z5) zu bestücken. Die acht niedrigwertigen Bits des Adreßbusses sowie der Datenbus sind durch die Bausteine Z16...Z18 gepuffert.

Z13 arbeitet als Speicher-Adreßdecoder und kann bis zu acht Speicherblöcke zu jeweils 2 KByte adressieren. Das EPROM sowie jeder RAM-Block belegen jeweils einen Bereich, so daß sich folgende Speicheraufteilung ergibt:

2400H	FFFFH	frei		
2000H	23FFH	RAM 2	Z2, Z3	1 KByte
1400H	1FFFH	frei		
1000H	13FFH	RAM 1	Z4, Z5	1 KByte
800H	FFFH	frei		
0H	7FFH	EPROM	Z10	2 KByte

Obwohl die Blockadressierung in Schritten von 2 KByte vorgenommen wird, liegt zwischen EPROM und dem ersten RAM ein Bereich von 2 KByte. Auf diese Weise ist es sehr einfach möglich, statt des EPROMs 2716 den Typ 2732 (mit doppelter Kapazität) zu verwenden (+5-V-Leitung zu Anschluß 21 auf der Platine auftrennen und Anschluß 21 an Adresse 11 legen).

Die Platine arbeitet mit einer Taktfrequenz von 2,4576 MHz, die durch den Baustein Z11 erzeugt wird. Diese Frequenz erlaubt einen besonders einfachen Aufbau des Baud-Rate-Generators: Ein Binärteiler (Z12) stellt die benötigten Frequenzen an Steckbrücken zur Verfügung:

Z12-Anschluß	Steckbrücke	Frequenz
13	1	153,6 kHz
11	2	76,8 kHz
10	3	38,4 kHz
9	4	19,2 kHz
8	5	9,6 kHz

Durch Wahl einer der fünf Brücken wird die entsprechende Frequenz an die Sende- und Empfangsregister der UARTs (Z8 und Z9) gelegt. Die Baud-Rate jedes Kanals ergibt sich aus dieser Frequenz sowie einem Teilerfaktor, der per Software programmiert werden kann. Dieser Faktor kann 1, 16 oder 64 sein, so daß die Baud-Rate im Bereich von 150 Bd (= 9,6/64) bis 153 600 Bd (= 153 600/1) liegen könnte. Erlaubt sind jedoch nur Geschwindigkeiten bis 9600 Bd (*Tabelle 1*). In den meisten Fällen bietet es sich an, den Faktor 16 zu programmieren, da damit die üblichen Geschwindigkeiten einstellbar sind. Durch Wahl unterschiedlicher Faktoren oder durch Einbau eines zweiten Satzes von Steckbrükken sind unterschiedliche Baud-Raten für beide UARTs möglich.

Ein-/Ausgabe-Möglichkeiten

Z6 und Z7 arbeiten als PIO-Bausteine und stellen dem Anwender insgesamt bis zu 48 TTL-kompatible Leitungen zur Verfügung, die als Ein- oder Ausgang zur Steuerung von Relais usw. bidirektional oder mit Handshake-Signalen benutzt werden können. Beim Einschalten der Versorgungsspannung werden diese Bausteine durch das aus R9 und C8 bestehende RC-Glied zurückgesetzt.

Der Baustein Z15 erlaubt die Abfrage von sechs DIL-Schaltern per Programm. Ein geschlossener Schalter legt das an diesem Eingabekanal liegende Bit auf „0", bei geöffnetem Schalter liegt über Pull-up-Widerstände „1" an.

Z14 arbeitet als Adreßdecoder für sämtliche Ein-/Ausgabe-Bausteine. Da die niedrigstwertigen zwei Adreßbits von den PIO-Bausteinen für interne Adressierungen verwendet werden, decodiert dieser Baustein die Adressen 2...4. Von den damit möglichen acht I/O-Adressen sind sechs belegt. Sie ergeben zusammen mit der internen Decodierung der PIOs und UARTs den in *Tabelle 2* dargestellten Ein-/Ausgabe-Bereich.

Alle Ein-/Ausgabe-Signale sowie die Versorgungsspannung sind an eine 64polige VG-Leiste geführt. An dieser Leiste können die anwenderspezifische Schaltung sowie das Netzteil aufgesteckt werden (*Tabelle 3*). Die Bezeichnungsweise der einzelnen Signale entspricht weitestgehend der in den Datenbüchern üblichen Art. Da die Bausteine 8255 und 8251 jeweils zweimal vorhanden sind, gibt eine nachgesetzte Ziffer an, um welchen es sich handelt (/1 = Z6 bei PIO-Signalen; Z8 bei UART-Signalen; /2 = Z7 bei PIO-Signalen; Z9 bei UART-Signalen).

Die Funktion der Stifte a4 und c4 ist nicht festgelegt. Je nach Anwendung

Tabelle 1: Die Baud-Rate der UARTs ergibt sich aus Taktfrequenz und programmiertem Faktor

Brücke	Faktor	Baud-Rate
1	1	nicht benutzen
1	16	9600 Baud
1	64	2400 Baud
2	1	nicht benutzen
2	16	4800 Baud
2	64	1200 Baud
3	1	nicht benutzen
3	16	2400 Baud
3	64	600 Baud
4	1	nicht benutzen
4	16	1200 Baud
4	64	300 Baud
5	1	9600 Baud
5	16	600 Baud
5	64	150 Baud

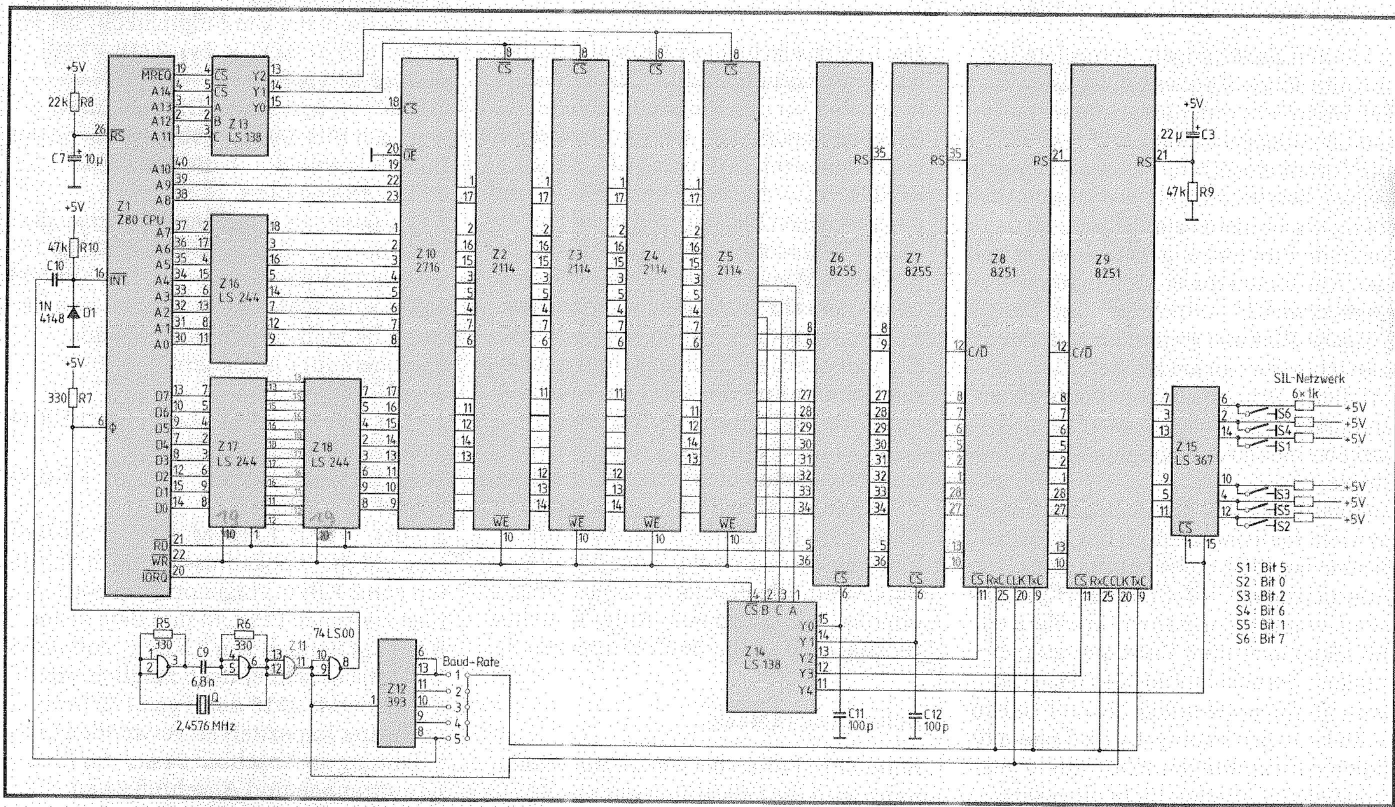


Bild 1. Schaltung des Z80-Einplatinencomputers

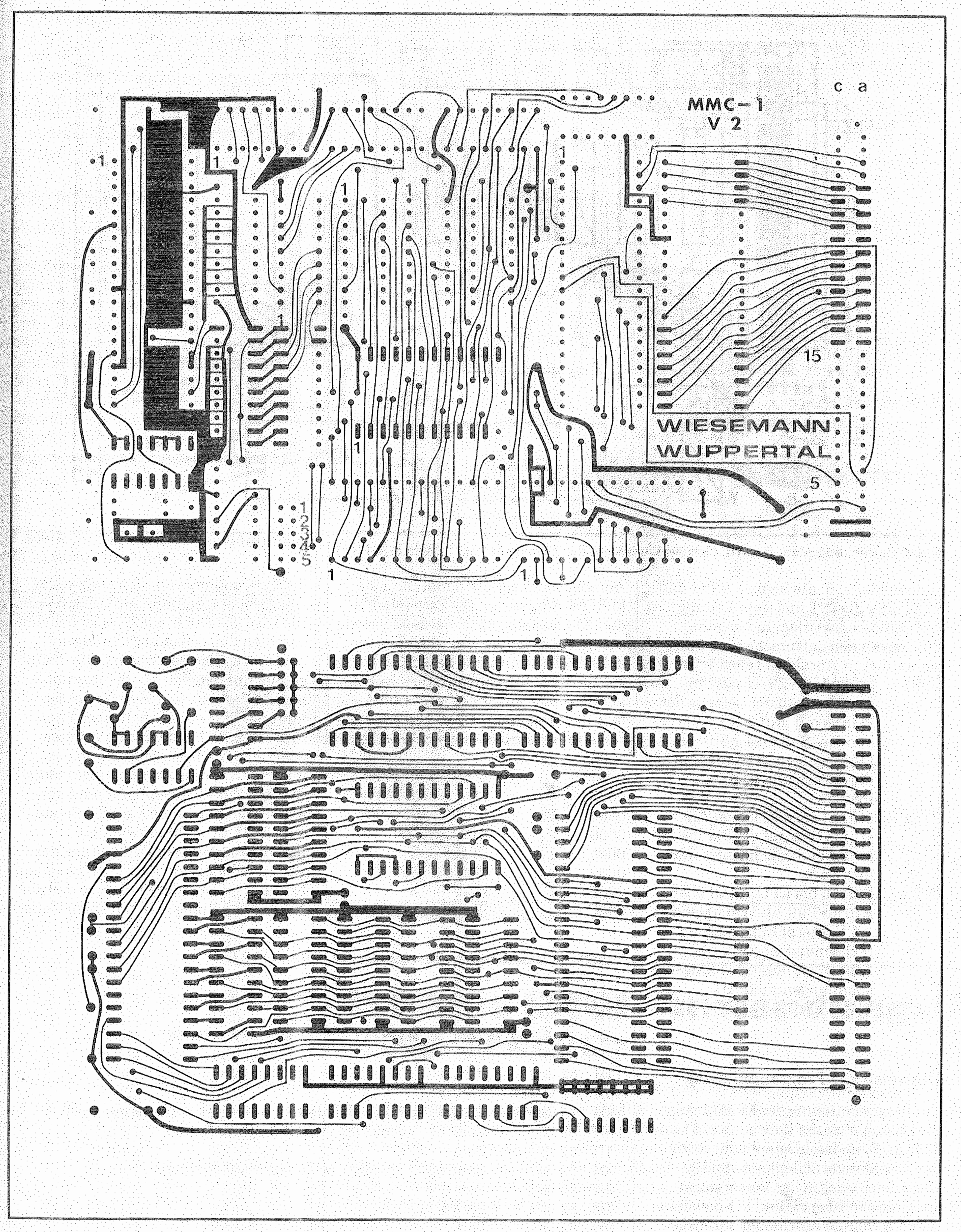


Bild 2. Platinenlayout

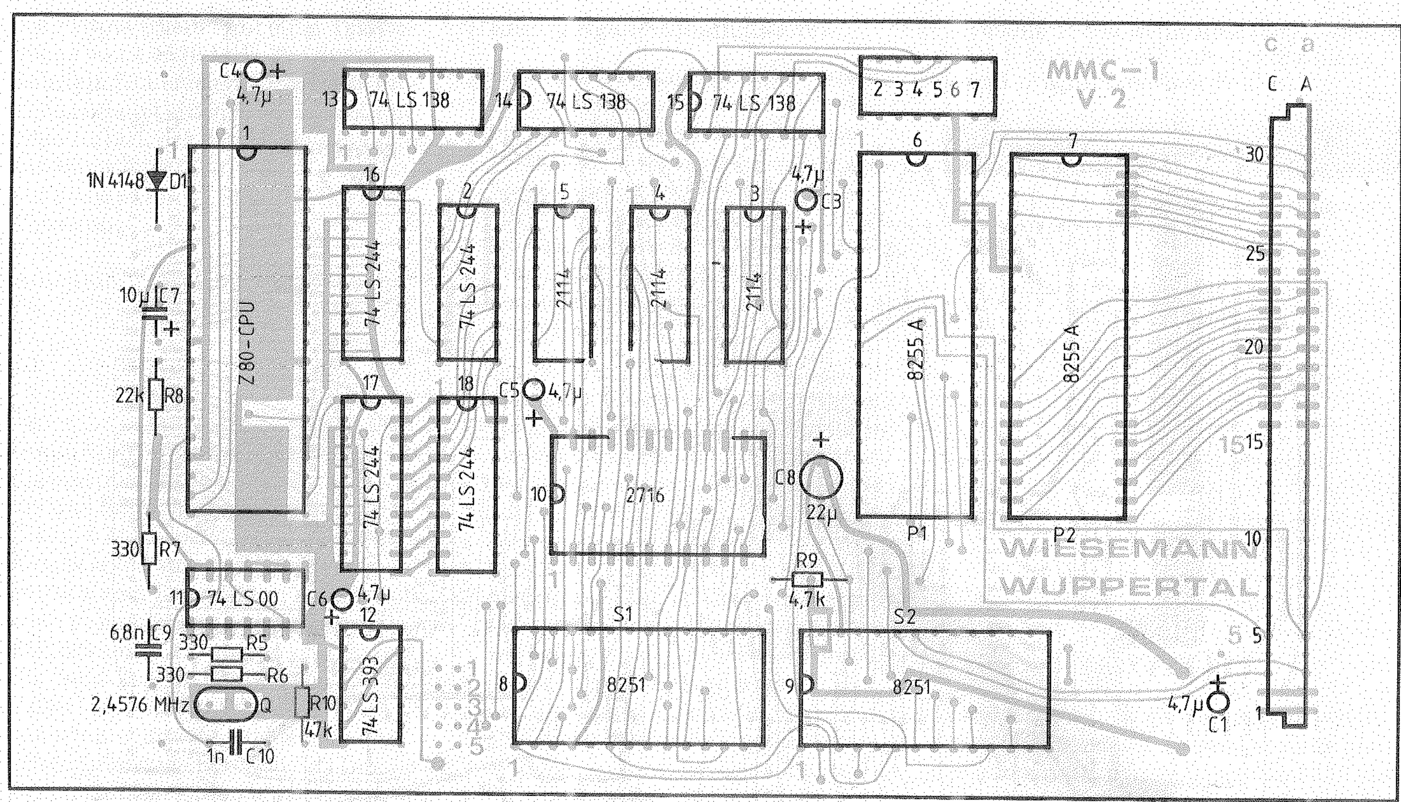


Bild 3. Bestückungsplan. Das SIL-Netzwerk wird auf der Lötseite bestückt (Switch-In-Line)

können hier z. B. die Signale $\overline{\text{DSR1}}$ und $\overline{\text{RTS1}}$ oder die $\overline{\text{INT}}$ und Takt-Leitung angeschlossen werden, indem eine Brücke von den entsprechenden Punkten an diesen Anschluß gelegt wird:

$\overline{\text{DSR1}}$ = Stift 22 von Z8. Dieser Anschluß ist auf der Lötseite der Platine mit Stift 26 (+5 V) verbunden. Soll dieser Stift als Eingang benutzt werden, so ist diese Verbindung aufzutrennen.

$\overline{\text{RTS1}}$ = Stift 23 von Z8. Dieser Anschluß liegt auf der Platine frei, so daß er direkt benutzt werden kann.

$\overline{\text{INT}}$ = Stift 16 der CPU – liegt über R10 als Pull-up-Widerstand an +5 V. Dieser Eingang kann direkt benutzt werden. Als Schutz vor negativen Eingangsspannungen dient D1.

Takt = Stift 1 von Z12 (2,4576 MHz).

Hinweise zur Programmierung

Zur Programmierung des MMC-1 sind die Datenblätter der Bausteine 8251 und 8255 (z. B. im Intel-Datenbuch) sowie das Z80-Manual (Zilog) notwendig. Nach dem Anlegen der Versorgungsspannung werden µP und I/O-Bausteine durch getrennte RC-Glieder zurückgesetzt. Der µP legt Adresse 0 an den Adreßbus und spricht damit die erste EPROM-Adresse an. Die Parallelports der I/O-Bausteine 8255 werden in Mode 0 (Input) geschaltet. Um zu vermeiden, daß der µP (wenn seine Reset-Zeit kürzer als die der I/O-Bausteine ist) zu früh die I/O-Bausteine anspricht, sollte am Anfang des Programms eine Verzögerungsschleife vorgesehen werden. Zum Beispiel:

```
0000          LD     A, 08H
0002    V1    LD     C, 0FFH
0004    V2    LD     B, 0FFH
0006    V3    DJNZ   V3
0008          DEC    C
0009          JR     NZ, V2
000B          DEC    A
000C          JR     NZ, V1
```

Bei Verwendung von Interrupts sollte zuvor der Stackpointer initialisiert sein. Die Lage des Stacks im RAM ist beliebig; beachten Sie, daß er in Richtung auf niedrigere Adressen wächst. Die Platine erlaubt nur den Betrieb im Interrupt-Mode 1 des Z80. In dieser Betriebsart führt die CPU nach Erhalt eines Interrupt einen RST-Befehl nach Adresse 038H aus. An dieser Stelle kann das Interrupt-Unterprogramm programmiert werden. Zur Rückkehr aus dem Interrupt-Programm reicht ein RET-Befehl aus (RETI ist nicht notwendig).

Die Verwendung der $\overline{\text{INT}}$-Leitung des Z80 ist beliebig (ein Pull-up-Widerstand befindet sich auf der Platine, so daß eine externe Schaltung z. B. mit Open-Collec-

Tabelle 2: Ein-/Ausgabe-Bereich der PIOs und UARTs

15H	8251/2	Control	(8251/2 = Z9 auf MMC-Platine)
14H	8251/2	Data	
13H	8255/2	Control	(8255/2 = Z7 auf MMC-Platine)
12H	8255/2	Port C	
11H	8255/2	Port B	
10H	8255/2	Port A	
8H	74LS367	(Z15)	(DIL-Schalter 1...6) – diese Adresse nur lesen
5H	8251/1	Control	(8251/1 = Z8 auf MMC-Platine)
4H	8251/1	Data	
3H	8255/1	Control	(8255/1 = Z6 auf MMC-Platine)
2H	8255/1	Port C	
1H	8255/1	Port B	
0H	8255/1	Port A	

tor-Ausgang angeschlossen werden kann) – diese Leitung kann über eine kurze Brücke an einen der freien Stifte auf der VG64-Leiste gelegt werden. Wird im Programm eine Zeitbasis benötigt, so kann durch Einbau von C10 ein Interrupt-Signal alle 256 Taktzyklen erzeugt werden (alle 0,1042 ms).

Programmbeispiele

Einige Unterprogramme können in vielen Anwendungen des MMC-1 unverändert übernommen werden. Alle nachfolgenden Unterprogramme (*Bilder* 4 und 5) kehren mit gesetztem Z-Flag zurück, wenn die entsprechende Ein- oder Ausgabe nicht möglich war (Eingabe: Es ist kein Zeichen empfangen worden; Ausgabe: Das zuvor gesendete Zeichen ist von der Peripherieschaltung noch nicht abgerufen worden). Auszugebende Zeichen müssen in Register C liegen, empfangene Zeichen gelangen in Register C.

Tabelle 3: Signale an der 64poligen VG-Leiste des MMC-1

	Reihe a	Reihe c
1	GND	GND
2	+5 V	+5 V
3	DSR2	RTS2
4	sh. Text	sh. Text
5	PC7/1	PC6/1
6	PC4/2	DTR2
7	DTR1	CTS1
8	CTS2	T × D 2
9	T × D 1	R × D 2
10	R × D 1	PB2/1
11	PB1/1	PB0/1
12	PC3/1	PC2/1
13	PC0/1	PC4/1
14	PC5/1	PC1/1
15	PB3/1	PB4/1
16	PB5/1	PB6/1
17	PB7/1	PB3/2
18	PB4/2	PB5/2
19	PB6/2	PB7/2
20	PB2/2	PC5/2
21	PB1/2	PB0/2
22	PC3/2	PC2/2
23	PC1/2	PC0/2
24	PC6/2	PC7/2
25	PA7/2	PA0/2
26	PA6/2	PA1/2
27	PA5/2	PA2/2
28	PA4/2	PA3/2
29	PA6/1	PA7/1
30	PA5/1	PA0/1
31	PA1/1	PA4/1
32	PA2/1	PA3/1

Die Programme verändern den Inhalt der Register A und F. Vor Aufruf muß der entsprechende Baustein für diese Betriebsart vorbereitet – d. h. ein PIO-Baustein muß zuvor den entsprechenden Betriebsartbefehl, ein UART-Baustein den Betriebsartbefehl und ein Kommando erhalten haben.

```
;Eingabe mit Strobe über Port A in Z6
PA1IN       IN      A,(PC1)
            BIT     5,A             ;Bit 5 = 1, wenn Daten
                                    ;empfangen wurden
            RET     Z               ;Z-Flag, wenn keine Daten
            LD      C,A             ;Sonst Rückkehr mit Daten
            RET                     ;in Register C

;Ausgabe mit Strobe über Port B in Z6
PB1OUT      IN      A,(PC1)
            BIT     1,A             ;Bit 1 = 1, wenn Daten
                                    ;gesendet werden dürfen
            RET     Z               ;Wenn nicht, zurück mit
                                    ;Z-Flag
            LD      A,C             ;Sonst Daten aus Register
                                    ;C ausgeben
            OUT     (PB1), A
            RET
```

Bild 4. Unterprogramme für die parallele Ein-/Ausgabe

```
;Serielle Eingabe über Z8
S1IN        IN      A,(SC1)
            BIT     1,A             ;Bit 1 = 1, wenn Daten
                                    ;empfangen wurden
            RET     Z               ;Wenn nicht, zurück mit
                                    ;Z-Flag
            IN      A,(SD1)         ;Sonst Daten einlesen
            RET

;Serielle Ausgabe über Z8
S1OUT       IN      A,(SC1)         ;Bit 0 = 1, wenn Daten
            BIT     0,A             ;ausgegeben werden
                                    ;können
            RET     Z               ;Wenn nicht, zurück mit
                                    ;Z-Flag
            LD      A,C             ;Sonst Daten aus Register
            OUT     (SC1), A        ;C ausgeben
            RET
```

Bild 5. Unterprogramme für die serielle Ein-/Ausgabe

Ist Ihre EMUF-Anwendung „top secret"?

Der in mc 1981, Heft 2, beschriebene „Einplatinen-Mikrocomputer für universelle Festprogramm-Anwendung" (EMUF) eignet sich für tausend Dinge – oder sogar noch mehr. Vielleicht realisieren Sie mit der EMUF-Platine gerade eine allgemein interessierende Anwendung? Rufen Sie uns an oder schreiben Sie uns! Wir freuen uns über jede neue EMUF-Idee, und vielleicht sind andere Leser ganz gierig darauf, Ihre Anwendung kennenzulernen. Oder ist Ihre EMUF-Applikation etwa „top secret"? *Ihre mc-Redaktion*

Karl-Heinz Sergel

CP/M – eine Sache mit Zukunft

CP/M ist sicherlich das am weitesten verbreitete Betriebssystem für 8-Bit-Rechner mit einem 8080/Z80/8085-Prozessor. Seine Entwicklung, seinen Aufbau und seine Fähigkeiten sollen in diesem Aufsatz dargestellt werden. Da jetzt viele der großen Firmen mit CP/M-Computern auf den Markt kommen werden, ist es nicht zu gewagt, zu behaupten, daß CP/M eine Sache mit Zukunft ist.

Vater des CP/M ist Gary A. Kindall. 1973 war er als Softwareentwickler bei Intel tätig und damit beschäftigt, für das Intellec-8-Entwicklungssystem einen PL/M-Compiler zu entwickeln. Masatoshi Shima hatte den ersten lauffähigen 8080-Mikroprozessor fertig, und die von IBM entwickelten Floppy-Disks schienen es möglich zu machen, auf Lochstreifen als Massenbackup bei Kleincomputern zu verzichten. Kindall bekam von Shugart einen der ersten Testdrives zur Verfügung gestellt und zusammen mit John Torode baute er Anfang 1974 einen ersten Diskcontroller zusammen. Ein File-Handling-System war ebenfalls von Kindall in der Zwischenzeit geschrieben worden und nach dem Laden des Maschinencodes über den Lochstreifenleser erschien 1974 der erste CP/M-Prompt.
Ein erster kommerzieller Einsatz dieses neuen Systems fand 1975 statt, aber CP/M (Control Program for Mikroprocessors) fand nur wenig Aufmerksamkeit. Kindall schrieb in dieser Zeit weitere Hilfsprogramme für sein CP/M: einen Editor, einen Assembler und einen Debugger, Vorläufer der heutigen ED-, ASM- und DDT-Programme. Anfang 1976 war CP/M vier verschiedenen Disk-Controllern angepaßt. Als Mitte 1976 die Imsai-Corporation eine Anpassung an ihr System benötigte, nahm Kindall eine weitgehende Umstrukturierung des gesamten CP/M vor, um die Arbeit bei Neuanpassungen zu reduzieren. Alle hardwareabhängigen Teile wurden in einem besonderen Modul dem BIOS (Basic Input/Output System), zusammengefaßt. Damit war mit der Version 1.3 die Struktur hergestellt, die das CP/M auch heute noch besitzt.

Der Autor ist Mitarbeiter der Heath GmbH.

Das ist CP/M

CP/M ist ein fileorientiertes Betriebssystem. Ein File bei CP/M ist eine Zusammenfassung von Daten beliebiger Länge, die vom Anwender mit einem beliebigen symbolischen Namen von acht Zeichen und einer dreistelligen Bezeichnung des Filetyps versehen werden können. Es kann sich bei einem solchen File um ein lauffähiges Programm (Filetyp .COM), um ein Programm im Intel-Hexformat (Filetyp .HEX), um einen Basic-Sourcecode (Filetyp .BAS) o. ä. handeln. Die Aufzeichnung und das Wiederfinden eines solchen Files spielt sich für den Anwender lediglich im Aufruf der entsprechenden Funktionen des Betriebssystems ab. Mit der tatsächlichen physikalischen Verteilung auf der Diskette hat der Anwender aber nichts mehr zu tun. Das eigentliche Betriebssystem besteht aus drei Modulen: dem CCP, dem BDOS und dem BIOS. Hinzu kommen Hilfsmittel, die jedoch nur bei Bedarf von der Diskette in das User-RAM geladen werden. Als Grundlage der Beschreibung des CP/M soll die Version 2.2 dienen.

Der Console Command Prozessor „CCP"

Nach dem Bootstrapvorgang (dem Laden des Betriebssystems) übernimmt der CCP die Kontrolle über das System. Er meldet sich mit dem CP/M-Prompt A> und ist bereit für die Eingabe von Befehlen. Es können nun Programme geladen und zur Ausführung gebracht werden. Dies geschieht durch die Eingabe des Laufwerkes und des Filenamens. Befindet sich das Programm auf dem im Prompt angegebenen Laufwerk, so kann die Laufwerksangabe entfallen. Das aufgerufene Programm muß die Filebezeichnung .COM besitzen.

Beispiel:
A> B:TEST <disk:Filename>

Das CCP sucht nun auf der Diskette in Laufwerk B nach dem File TEST.COM. Wird es nicht gefunden, so antwortet CCP mit TEST ?.
Wird das Programm gefunden, so lädt CCP das Programm ab hex 100 in das User-RAM und bringt es durch einen Sprung nach hex 100 zur Ausführung. CCP verfügt über die folgenden sechs Befehle, die in *Tabelle 1* aufgelistet sind. Für die Funktion des BDOS ist der CCP entbehrlich. Von einem Anwenderprogramm oder einem Basic-Interpreter kann er ohne Funktionseinbuße für das Betriebssystem überschrieben werden, er wird bei jedem Warmstart des Systems neu geladen.

Das Basic Disk Operating System (BDOS)

Kern des hardwareunabhängigen Teils von CP/M ist das BDOS. Es stellt 37 verschiedene Funktionen zur Verfügung, die alle von Benutzerprogrammen aus aufrufbar sind. Die Funktionen 1 bis 12 bedienen die Konsole, den Drucker, den Lochstreifenleser und -stanzer. Bei den anderen Funktionen handelt es sich um Disketten-Ein- und -Ausgabebefehle. Die Universalität von CP/M liegt in der genauen stets gleichen Definition der Schnittstelle vom Benutzer zum BDOS. Die Anpassung an die jeweilige Maschine über das BIOS hat hierauf keinen Einfluß. Die Benutzung einer BDOS-Funktion läuft prinzipiell nach immer dem gleichen Schema ab und besteht aus drei Teilen (die Register sind die der 8080-CPU):

1. Das C-Register wird mit der Nummer der aufzurufenden BDOS-Funktion geladen.
2. Je nach Art der aufzurufenden Funktion wird das DE-Registerpaar mit einer Adresse bzw. einem auszugebenden Zeichen geladen.
3. Es erfolgt ein BDOS-Call (Einsprungpunkt ist hex 0005).

Nach der Ausführung enthält je nach Art der aufgerufenen Funktion entweder das A-Register einen Wert oder das HL-Registerpaar eine Adresse (Beispiele folgen später).
Weiterhin überprüft das BDOS jede Eingabe von der Konsole und erkennt die Zeichen <ctrl. C>=WARM START und <ctrl.P>= Drucker ein/aus.

Das Basic Input/Output System (BIOS)

Hierbei handelt es sich um den hardwareabhängigen Teil des CP/M Betriebssystems. Er nimmt für jedes Computersystem die Anpassung an die spezifische Systemkonfiguration vor. Bedingung an das BIOS ist, daß es mit einer Sprungtafel beginnt, die 17 Routinen (*Tabelle 2*) bedient (ob sich nun wirklich die entsprechenden Routinen dort befinden ist egal, zum Beispiel wird in mc 1980, Heft 3, die Verwendung der READER-Routine für einen Barcodeleser beschrieben).

CP/M-Hilfsmittel

Mit dem CP/M-System werden eine Anzahl von Hilfsmitteln bereitgestellt, die je nach Bedarf von der Diskette geladen werden können. Diese können von Computersystem zu Computersystem variieren, standardmäßig sind sie jedoch vorhanden (*Tabelle 3*):

Die Speicherorganisation

Bild 1 gibt einen Einblick in die Speicherbelegung durch CP/M. Die drei Module sind von der höchsten RAM-Adresse abwärts lokalisiert. Die wichtigsten Adressen sind FBASE, der Beginn des BDOS und die TBASE, der Beginn der „Transient Program Area", des für Anwenderprogramme zur Verfügung stehenden RAMs. Die ersten 256 Byte, die Zero-Page, sind für den Gebrauch durch CP/M freigehalten. Es befinden sich hier (alle Adressen hex):

0–2	Jump WARMBOOT
3	Das IOBYTE (Erklärung später)
4	Das momentan angesprochene Laufwerk (ab Null zählend)
5–7	Jump BDOS (dies ist der Einsprungspunkt für alle BDOS-Operationen, Byte 6 und 7 geben die FBASE wieder, FBASE-1 gibt die größte für Anwenderprogramme nutzbare RAM-Adresse an)
8	RST 1 Ansprungpunkt
10	RST 2
18	RST 3
20	RST 4
28	RST 5
30	RST 6
38	RST 7 wird vom DDT-Programm benutzt. Alle anderen Restartadressen sind momentan nicht benutzt.
5C–7C	Vom CCP zur Speicherung des FCB (File Control Block) benutzt.
80–FF	Vom CCP als File In-/Output-Buffer benutzt (DMA-Buffer).

Der genaue Wert der FBASE ist von der jeweiligen Größe des BIOS und des zur Verfügung stehenden User-RAMs abhängig. Bei dem von Digital Research vertriebenen CP/M für das Intel MDS-Intellec-System ist ein Bereich von 1536 Byte freigehalten, die FBASE liegt hier also bezogen auf 64 KByte User RAM bei EC00. Das Heath/Zenith 2.2.02 CP/M hat ein BIOS von 5 KByte, hier liegt – wieder bezogen auf 64 KByte – die FBASE bei DC06. Stets gleich ist die Länge des CCP (800H) und des BDOS (E00H).

Die Diskettenorganisation

CP/M ist für eine Vielzahl von Diskettenformaten, Aufzeichnungsformen und Kontrollern erhältlich bzw. anpaßbar. Zur Darstellung der Diskettenorganisation soll als Beispiel eine 8-Zoll-Diskette, einfache Schreibdichte, softsektoriert, dienen. Die 77 Spuren einer solchen Diskette sind in 26 Sektoren mit je 128 Byte Datenspeicherkapazität gegliedert. Dies entspricht dem IBM 3740 Diskettenstandard und solche Disketten heißen deshalb „IBM-kompatibel". Das Speichervermögen einer solchen Diskette beträgt also 77·26·128=256 256 Byte. Jede CP/M-Diskette besitzt drei verschiedene Bereiche. Spur 0 und Spur 1 sind für das CP/M-System reserviert, unabhängig davon, ob es sich überhaupt auf der Diskette befindet.

Spur 0 Sektor 1	Cold Start Loader
Spur 0 Sektor 2 bis Spur 0 Sektor 17	CCP
Spur 0 Sektor 18 bis Spur 1 Sektor 19	BDOS
Rest	BIOS

Den zweiten Bereich bildet die File Directory Area. Sechzehn Sektoren der zweiten Spur sind stets für das Directory reserviert. Dieser Bereich kann jedoch durch entsprechende Änderungen im BIOS erweitert werden. Der Rest der Diskette wird zur Speicherung von Daten und Programmen benutzt.

Datentransfer von und zur Diskette geschieht in Blöcken zu je 128 Byte. Logisch aufeinander folgende Fileblöcke werden jedoch nicht in entsprechender physikalischer Reihenfolge auf der Diskette abgelegt, da der Disk-Controller nach einem jeden Block erst die Richtigkeit der Übertragung prüfen muß und auch das BDOS einige Zeit bis zur nächsten Lese/Schreiboperation benötigt. Der Standardversatz beträgt sechs Sektoren (ebenfalls IBM-kompatibel), kann jedoch auch anders definiert werden. Die Zuordnung der logischen zur physikalischen Position geschieht über die schon erwähnte BIOS-Funktion 17.

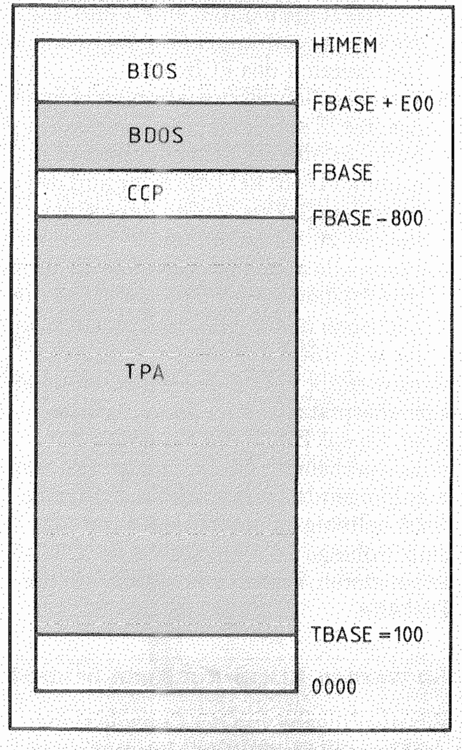


Bild 1. So liegt CP/M im Speicher

Die Verwaltung des Speicherplatzes auf der Diskette erfolgt unter CP/M dynamisch, das heißt, daß zum Zeitpunkt der Neueröffnung eines Files dessen mögliche Länge noch nicht angegeben werden muß. Speicherplatz wird einem File in Portionen zu je 1 KByte zugeordnet. Die Mittel, die CP/M für diese Diskettenverwaltung benötigt, sind:

1. Der File Control Block (FCB)
2. Die Allocation Bit Map (ABM)
3. Das Directory

Ein File Control Block benötigt 36 Byte RAM. *Bild 2* zeigt ihn schematisch. Seine einzelnen Teile haben folgende Bedeutung (*Tabelle 4*):

Die bereits erwähnten 16 Sektoren der zweiten Spur einer jeden CP/M-Diskette enthalten das vom BDOS verwaltete File Directory. Es enthält die jeweils ersten 32 Byte der zu den auf der Diskette befindlichen Files gehörenden File Control Blocks. Bei jedem Neuaufruf einer Diskette liest das BDOS diese 16 Sektoren und berechnet aus den Einträgen in dem DM.-Bereich die Belegung der Diskette. Diese Belegung faßt das BDOS zu der Allocation Bit Map zusammen und legt diese im RAM ab. Diese „Karte" besteht aus 243 Bit und spiegelt die Belegung der Diskette wieder.

Der Ablauf einer Fileeröffnung durch das BDOS auf der Diskette läßt sich nun vollständig beschreiben.

1. Das BDOS durchsucht die Allocation Bit Map, bis es eine 0 findet.

2. Es ersetzt die 0 durch eine 1 und trägt die Gruppennummer (1–243) in den DM.-Bereich des FCB ein.
3. Vor jeder Schreiboperation berechnet das BDOS aus der letzten Gruppennummer und der nächsten Recordnummer (Feld RC.) die Spur und den logischen Sektor, wohin der nächste Record zu schreiben ist.
4. Wenn alle acht Sektoren einer Gruppe beschrieben sind, sucht das BDOS die nächste 0 in der ABM und macht weiter wie unter 2.

Daraus ergibt sich, daß das Minimum an Speicherplatz für ein File 1 KByte beträgt. Da das BDOS die ABM stets von vorn durchsucht, können logisch zusammenhängende Teile eines Files physikalisch beliebig auf der Diskette verteilt sein. Solange freie Blöcke auf der Diskette existieren, schreibt das BDOS auf die Diskette.

So werden BDOS-Routinen benutzt

Für die Eingabe von der Console stehen drei verschiedene BDOS-Routinen zur Verfügung. Als Beispiel diene die Funktion 10: Read Console Buffer.

```
RECOBU MVI C,0AH    ; Funktion 10
                      soll ausgeführt
                      werden
       LXI D,02000H; Buffer beginnt
                      bei 2000
       MVI A,020    ;
       STAX D       ; max 20
                      (dezimal)
                      Zeichen sollen
                      eingelesen
                      werden
       CALL 05H     ; BDOS Call
       (Delimiter ist <return>)
```

Bei der Rückkehr von dieser Funktion enthält 2000 die 20, 2001 enthält die Anzahl der eingelesenen Zeichen. Es folgt die durch 2001 wiedergegebene Anzahl eingelesener Zeichen.

Um auf einen bereits existierenden File zugreifen zu können, muß er geöffnet werden:

a) 36 Byte USER-RAM für einen FCB müssen bestimmt werden.
b) Die Felder FN. und FT. werden mit dem Filenamen und dem Filetype gefüllt.
c) Das C-Register wird mit 0F (BDOS Funktion (15) OPEN FILE) geladen.
d) Das DE-Register wird mit der Adresse des FCB geladen.
e) Call BDOS.
f) Das A-Register enthält OFFH, wenn der File nicht gefunden wurde, oder die Nummer des Directory-Eintrags.

Die Startadresse des Buffers, aus dem oder in den Daten auf oder von der Diskette geschrieben werden, wird DMA (Direct Memory Access) genannt. Die Minimalgröße dieses Buffers beträgt 128 Byte. Wenn er nicht näher bestimmt wird, nimmt BDOS hex 080 als DMA an.

a) Das C-Register mit hex 19 laden (BDOS-Funktion 26).
b) Das DE-Registerpaar mit der Startadresse laden.
c) Call BDOS.

Nach dem Öffnen einer Files muß für jeden Record ein gesonderter Lese- bzw. Schreibbefehl erfolgen. Der Datentransfer findet über den letzten gesetzten DMA-Buffer statt.

a) Laden des C-Registers mit dem Funktionscode (21 = Write Next Record, 20 Read Next Record).
b) Das DE-Registerpaar mit der FCB-Adresse laden.
c) Call BDOS.

Nach Ausführung der Lese-/Schreibfunktion erhöht BDOS den Recordzähler im NR.-Feld des FCB und kommt mit einer Ausführungsmeldung im Register A zurück.

Inhalt A (dezimal)	Nach Lesen	Nach Schreiben
0	Lesen erfolgreich	Schreiben erfolgreich
1	Leseversuch nach EOF	Fehler beim Versuch, einen Fileextent zu öffnen
255	Fehler	Kein Platz mehr im Directory

Wie bereits im Abschnitt „Diskettenorganisation“ angedeutet, sind für das BIOS auf den beiden Systemspuren 7 Sektoren frei. Für ein kleines BIOS mag das ein ausreichender Platz sein, um jedoch die Anpassungsmöglichkeiten, die CP/M gerade so attraktiv machen, voll zu nutzen, reicht er nicht aus. Es gibt nun mehrere Möglichkeiten, das System mit einem größeren BIOS zu versehen.

Bei Heath/Zenith und dem zugehörigen CP/M 2.2 wurde zum Beispiel auf eine Unterbringung der BIOS in den Systemspuren überhaupt verzichtet. Dafür wurde der Cold Start Loader durch einen sogenannten Bios Loader ersetzt, der zuerst das BIOS, welches als normales File auf der Diskette gespeichert ist, in den User-RAM lädt. Unter der Kontrolle des BIOS wird dann der Rest des Systems geladen. Diese Methode hat den Vorteil, daß durch bloßes Austauschen des BIOS-Files eine völlig andere Systemanpassung erreicht werden kann, ohne die Systemspuren überhaupt anzutasten.

Der Nachteil besteht darin, daß der BIOS-File relokatibel sein muß.

Im Abschnitt über die Speicherorganisation wurde bereits auf das IOBYTE hingewiesen. Mit seiner Hilfe läßt sich eine Zuordnung der vier logischen Devices (Console, Reader, Punch und Listener) zu im BIOS definierten physikalischen Einheiten erreichen. Ein entsprechendes Zuordnungsprogramm CONFIGUR.COM macht CP/M zu einem überaus flexiblen Betriebssystem.

Tabelle 1: Die sechs Befehle des CCP

REN <disk:filename.ext>=<disk:filename.ext>
Dieser Befehl gestattet es, beliebige Files umzubenennen.

ERA <disk:filename.ext>
Hiermit können Files auf der Diskette gelöscht werden.

DIR <disk:>
Es werden alle Files einer Diskette, die nicht das Systemfile Attribut besitzen, aufgelistet.

SAVE nn <disk:filename.ext>
Mit diesem Befehl kann ein beliebig langer Inhalt des USER-RAMs ab hex 100 in Blöcken zu je 256 Byte unter einem wählbaren Namen auf Diskette abgelegt werden.

TYPE <disk:filename.ext>
Diese Anweisung stellt den Inhalt eines (Text-)Files auf dem Konsolschirm dar.

USER nn
Unter CP/M, Version 2.2, kann der Speicherplatz auf der Diskette in bis zu 16 verschiedene Benutzerbereiche unterteilt werden. Mit dieser Funktion wird ein solcher Benutzerbereich aufgerufen.

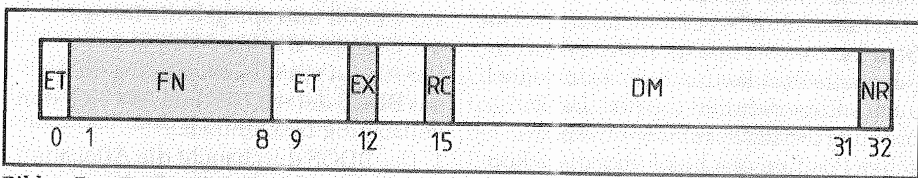


Bild 2. Das Format des File Control Blockes

Tabelle 2: Die 17 Unterprogramme, aus welchen BIOS besteht

Nr.	Unterprogramm	Beschreibung
1.	COLD BOOT	Nach dem Laden des Systems sorgt diese Routine für das Initialisieren von Systemparametern.
2.	WARM BOOT	Hiermit wird das BDOS und der CCP nach einem <ctrl.C> neu geladen.
3.	CONSOLE STATUS	Fragt die Konsole ab, ob eine Eingabe erfolgt ist.
4.	CONSOLE INPUT	Eingabe von der Konsole.
5.	CONSOLE OUTPUT	Ausgabe an die Konsole.
6.	LIST OUTPUT	Ausgabe an den Drucker.
7.	PUNCH	Ausgabe an die Stanze.
8.	READER	Eingabe vom Lochstreifenleser.
9.	HOME	Das gerade angesprochene Diskettenlaufwerk sucht die Spur 00.
10.	SELECT DISK	Ein Diskettenlaufwerk wird ausgewählt.
11.	SET TRACK	Eine bestimmte Spur der Diskette wird angewählt.
12.	SET SECTOR	ein bestimmter Sektor wird angewählt.
13.	SET DMA ADDRESS	Bestimmt den Bufferbereich im USER-RAM, von bzw. in den bei der nächsten Diskettenoperation geschrieben (gelesen) wird.
14.	READ	Einlesen von der Diskette in den DMA-Bereich, jeweils 128 Byte.
15.	WRITE	Ausgeben auf Diskette vom DMA-Bereich, 128 Byte.
16.	LIST STATUS	Abfrage ob der Drucker bereit ist.
17.	SECTOR	Übersetzung der logischen in die physikalische Position eines Sektors auf der Diskette.

Tabelle 3: Die standardmäßigen Utilities von CP/M

Utility	Beschreibung
ED.COM	Ein zeilenorientierter Texteditor.
ASM.COM	Ein 2-Pass-Assembler für Intel 8080 Standardformat ASCII-Files. Produziert Hex-Files im Intel-Hex-Format.
LOAD.COM	Wandelt Files im Intel-Hex-Format in lauffähige Maschinencode-Programme.
STAT.COM	Diskettenstatus-Hilfsmittel, enthält Routinen zur Ermittlung von freiem Speicherplatz, der Länge von Files, zum Setzen von I/O- und Systemattributen, usw.
DDT.COM	Debugger mit einer Vielzahl von Funktionen.
SUBMIT.COM	Ein Hilfsprogramm zur Stapelverarbeitung unter dem CCP.
XSUB.COM	Im Zusammenhang mit SUBMIT gestattet es die Stapelverarbeitung von Befehlseingaben in durch SUBMIT aufgerufene Programme.
PIP.COM	Ein Fileübertragungsprogramm.
SYSGEN.COM	Mit diesem Programm können die Systemspuren einer Diskette gelesen und beschrieben werden.
MOVCPM. COM	Dieses Programm ermöglicht die Anpassung von CP/M an verschiedene User-RAM-Größen.

Tabelle 4: Die Bedeutung der Bytes im FCB

Feld	Byte	Bedeutung
ET.	Byte 0	Enthält den Laufwerkscode (ab 0 zählend)
FN.	Byte 1–8	Enthält den linksbündigen Filenamen, wird mit hex 20 aufgefüllt.
FT.	Byte 9–11	Dieses Feld enthält den Filetype. Gewisse Programme verlangen nach bestimmten Typenbezeichnungen (ASM für ein Assembler Sourcefile, SUB für ein File zur Stapelverarbeitung usw.).
EX.	Byte 12	Ein FCB beschreibt ein Filestück von bis zu 16 KByte Länge. Dieses Feld enthält die Angabe, um das wievielte 16 KByte Stück eines Files es sich handelt.
	Byte 13–14	Werden nicht benutzt, enthalten 0.
RC.	Byte 15	Enthält die Anzahl der Records, die durch diesen FCB belegt werden.
DM.	Byte 16–31	Enthält die Nummern der 1 KByte Blöcke, auf denen sich der File (das Filestück) auf der Diskette befindet.
NR.	Byte 32	Dieses Byte wird nicht auf Diskette zurückgeschrieben, sondern vom BDOS während Lese/Schreiboperationen als Recordzähler benutzt.
	Byte 33–35	Werden für Random-Access-Zugriffe benötigt.

Ulrich Rohde

CS 2000 – ein preiswertes System für Profis

Wenn man sich genau überlegt, wie ein gut konzipierter Computer aussehen soll, dann kann man eigentlich nur zu einem Ergebnis kommen: Die Hardware sollte so beschaffen sein, daß möglichst viel frei programmierbarer Speicherplatz und eine allgemein akzeptierte CPU vorhanden sind, daß in einem zügig arbeitenden Massenspeicher die Daten aus dem Zentralspeicher dauerhaft und schnell zugänglich abgelegt werden können und daß die Bedienungselemente des Computers so geschaltet sind, daß bei keiner Manipulation ungewollt größere Datensätze gelöscht werden können.

Der CS 2000 erfüllt diese so lapidar hingeschriebenen Voraussetzungen gut. Er ist ein Z80-Rechner, der per Hardware darauf eingestellt ist, CP/M zu fahren. Das bedeutet, daß im Rechner ein Hard- oder Firmware-Baustein existiert, der ihn fähig macht, im Urzustand nach dem Einschalten ein Programm von der angeschlossenen Floppy-Disc-Einheit automatisch zu laden und sinnvoll in dieses geladene Programm einzuspringen. Beim CS 2000 ist dieses Urlade-System über mehrere Platinen verstreut.

S100: Hardware mit System

Der CS 2000 ist nämlich ein S100-System. Kenner wissen, daß solch ein System aus abgeschlossenen Karteneinheiten besteht, die untereinander über ein „Leitungsbündel", den S100-Bus, Daten und Steuersignale austauschen.

Es ist schon viel gelästert worden über diesen S100-Bus, der aus genau 100 Leitungen besteht, für welche genau festgelegt ist, was für Signale jeweils darauf liegen und wie das zeitlich-logische Zusammenspiel dieser Signale aussehen soll. Wahr ist zwar, daß das S100-Konzept ziemlich in der Anfangszeit der Mikrocomputerei entwickelt wurde und auch ein paar kritische Punkte enthält. Wahr ist aber auch, daß bis heute kein anderes Konzept existiert, das so weit standardisiert und so erprobt ist wie das S100-Bus-Konzept. Vor allem ist das mit den viel diskutierten Schwächen des S100-Bus so, daß sie höchstens dann zu Tage treten, wenn man superkritische Anwendungen in bezug auf die Hardware fahren will, dort wo man ohnehin besser nach militärischen Spezifikationen (mil. standards) arbeiten sollte.

Der CS 2000 der Firma Computershop besteht aus S100-Platinen von „California Computer Systems", die mit Netzteil und Gehäuse versehen werden. Standardmäßig besteht der Zentralrechner des CS 2000 aus einer CPU-Platine mit Z80A-Prozessor und RS-232-Schnittstelle sowie 2-KByte-Monitor: aus einer 64-KByte-Speicherplatine (dynamisch), die mit der 4-MHz-CPU zusammenspielen kann; dazu kommt noch eine Floppy-Disk-Controller-Platine, die bis zu



Die CS-2000-Konfiguration

vier Laufwerke von 5¼" oder 8" steuern kann; und eine In/Out-Platine, die die Kommunikation mit den Terminals und den Druckern erledigt. Das Urlade-Auto-Boot-System des CS 2000 besteht aus einem 2716-EPROM auf der Disk-Controllerplatine, in dem die Laderroutinen abgespeichert sind. Außerdem aus sogenannten Phantom-Steuerleitungen mit Verarbeitungslogik auf der CPU- und Speicherplatine. Und aus einem beliebig einstellbaren Systemstartvektor auf der CPU-Platine.

Hochfahren mit Bootstrap-Logik

Beim Einschalten sendet die CPU die Systemstartadresse aus, und wenn sie richtig gesetzt ist, dann antwortet das 2716-EPROM auf dem Floppy-Disk-Controller, wobei die Phantomlogik zugleich die RAM-Speicherplatine desaktiviert, damit nicht RAM und EPROM gleichzeitig Daten auf den Bus legen. Das Programm im EPROM bewirkt, daß von der Floppy-Station A (die erste, die immer geschlossen sein muß) das Betriebssystemprogramm in den Hauptspeicher geladen wird. Außerdem bewirkt es, daß nach erfolgreichem „Boot" des Systems in das Systemprogramm gesprungen wird. Dieses wiederum desaktiviert als erstes das EPROM, womit der Computer nun ungestört im RAM arbeiten kann. Geschildert wird das alles, damit sichtbar wird, daß doch recht komplexe Dinge ablaufen, wenn das System „hochgefahren" wird und weil hier besonders deutlich wird, daß der Computer sozusagen voll softwaredefinierbar ist. Wenn nämlich in Laufwerk A die CP/M-Systemdiskette eingelegt war, dann wird vollautomatisch ans obere Ende des RAM-Speichers das System geladen, das aus dem Command-Console-Prozessor (CCP), dem Basic Disk Operation System (BDOS) und dem Basic Input Output System (BIOS) besteht.

Die Peripherie des CS 2000

Die Programmodule des Systems sind vom Vertreiber des Computers auf die individuelle Konfiguration der Hardware und der Peripherie eingestellt worden. Das Modell, das bei uns stand, war mit dem Ampex-Dialog-80-Terminal ausgerüstet, einem recht intelligenten Gerät, dessen einziger Nachteil eine zu leicht verschiebbare Tastatur war. Leute mit schweren Fingern müssen die absetzbare Tastatur sicher irgendwie fixieren.
Als Drucker war bei uns der bewährte MX80 von Epson angeschlossen. Die Disk-Station bestand aus zwei Shugart-8"-Laufwerken mit Dual-Density-Format. Also standen 2,4 MByte Speicherplatz zur Verfügung.

Der Betrieb: klaglos

Das System stand etwa zwei Monate bei uns, und ich habe es immer dann, wenn nicht gerade ein Artikel zu schreiben war, kräftig sausen lassen (im wahrsten Sinne des Wortes: denn das Kühlgebläse kann man hören). Oftmals wurde es mehrmals an einem Tag ein- und ausgeschaltet. Zahllose Kalt- und Warmstarts von CP/M wurden durchgeführt. Alles hat das System klaglos mitgemacht. Die Floppy-Einheit arbeitete fehlerfrei. Die System-Disk hat allen Zerstörungs- und Abnutzungsversuchen widerstanden. Allerdings ist das Medium Floppy Disk so empfindlich, daß es immer empfehlenswert ist, in nicht zu weiten Zeitabständen Sicherheitskopien von der Arbeitsdiskette zu ziehen. Ein Punkt der Kritik hier ist (eine Eigenschaft fast aller vergleichbarer Systeme): Man kann durch Abschalten des Computers im falschen Moment bei eingelegter Diskette Datenverluste erzeugen. Und ich habe schon Profis gesehen, die versehentlich eine wichtige Diskette beim Abschalten im Laufwerk gelassen haben. Es ist dann immer sehr spannend, ob so ein Ereignis beim nächsten Lauf eine Fehlermeldung erzeugt oder nicht.

Weshalb einen CP/M-Rechner?

Das Schöne an jedem CP/M-Computer ist nämlich, daß man zum Beispiel mit einem Kommando namens SUBMIT eine ganze Folge von Programmen, die sich auf den Disketten befinden müssen, aufrufen und abarbeiten kann, als wären es einfache Befehle einer sehr mächtigen Computersprache. Man kann also Programmodule zu einem Paket zusammenpacken, einem Programm von Programmen. Das allerdings wird ein Benutzer, der erst mit CP/M beginnt, noch wenig ausnutzen. Solch einem Benutzer steht aber zum Beispiel die Welt aller Computersprachen offen. Die richtige Diskette eingelegt, verwandelt das Schlüsselwort CBASIC den Rechner in eine Basic-Maschine, deren Eigenschaften von wenigen Nur-Basicrechnern erreicht werden. Oder man benötigt einen Pascal-Computer: Einfach Diskette hinein, Pascal laden und fertig. Fortran, Cobol, Lisp und Forth, alles ist in ein paar Sekunden geladen.
Wer Anzeigen über Software studiert, weiß, daß zu jedem möglichen oder unmöglichen Problem auch eine Lösung existiert, die unter CP/M läuft. Beim Kauf einer solchen Lösung ist nur auf die richtige Formatierung der Diskette zu achten und darauf, daß der Speicherplatz des Computers ausreicht, denn CP/M läuft auf Rechnern mit Achtziger-CPUs ab 16 KByte RAM!
Ein vernünftiges Datenbankprogramm könnte man auf solch einem kleinen Rechner nicht fahren, da zuwenig Datensätze zugleich im Hauptspeicher behandelt werden könnten. Es gibt zum Beispiel einen Software-Leckerbissen im Programm, der von einer Berliner Firma (Severit Computer) hergestellt ist und der DEKOR heißt. Dieses Programm ist deshalb bemerkenswert, weil es ein wirklich universelles Datenbanksystem ist, das den Namen auch verdient. Mit ihm kann man schnell und flexibel eine Datei aufbauen, nach den verschiedensten Kriterien durchsuchen (zum Beispiel: Liste aller Kunden mit einem Umsatz größer als 1000 DM und mit Wohnort in Bayern) und mit wenig Aufwand pflegen und ändern. Neu ist an diesem Programm, daß der Benutzer selbst seine Datenstruktur einstellen kann und auch ohne Verluste später ändern kann, wenn die Umstände es erforderlich machen. Vor allem Kleinbetrieben ist damit ein Programmsystem an die Hand gegeben, das es dem Betrieb selbst erlaubt, die bedeutungsvollen Daten zu definieren und nicht durch externe EDV-Leute bestimmen zu lassen. Mit diesem Programm und dem CS 2000 hätte man dann ein recht schlagkräftiges kommerzielles Rechnersystem in der Hand, das seine Grenzen nur in der Geschwindigkeit besitzt, wenn mehr als 2000 Datensätze gleichzeitig verarbeitet werden sollen.

„Sehr" kompatibel

Das Wörtchen „sehr" soll dabei sagen, daß es schon noch Feinheiten geben kann, die ein unter CP/M laufendes Programm auf einer bestimmten Maschine zum Aussteigen zwingen. Und zwar vor allem, wenn bestimmte Hardwarevoraussetzungen nicht gegeben sind. Das aber ist selten. Ansonsten könnte man CP/M-Rechner gleicher Speicherkapazität nur noch nach Geschwindigkeit und Zuverlässigkeit unterscheiden. Und da ist der CS 2000 nicht zu beanstanden. Übrigens, die mitgelieferte Dokumentation war überwältigend: Vom CP/M-Handbuch (Zaks) bis zum Schaltplan einschließlich der wichtigsten Listings war alles im 10 cm hohen Bücherstapel vorhanden.

Herwig Feichtinger

Ein würdiger Nachfolger

Rockwell's AIM-65/40

Der Mikrocomputer AIM-65 hat sich vor allem als preiswertes 6502-Entwicklungssystem einen Namen gemacht. Viele seiner Beschränkungen wurden nun in einer größeren (und leider dreimal teureren) Version behoben: Der AIM-65/40 besitzt ein vierzigstelliges Display, einen breiteren Drucker und drei Mikroprozessoren.

Der AIM-65/40 soll die Lücke zwischen dem preiswerten Entwicklungssystem AIM-65, das inzwischen auch von mehreren OEM-Firmen als Komplettgerät im Gehäuse angeboten wird, und dem teureren „System 65" füllen. Gegenüber dem bekannten AIM-65 (Siemens vertreibt ihn als PC-100-Kit) besitzt die neue Version ein vierzigstelliges, grünleuchtendes 16-Segment-Fluoreszenz-Display mit Dezimalpunkten, die hier zur Kennzeichnung der sonst nicht darstellbaren Kleinbuchstaben dienen. Der Drucker bringt ebenfalls 40 Zeichen in eine Zeile, kann Kleinbuchstaben und zahlreiche Sonderzeichen (z. B. Indexziffern, Wurzel u. v. a.) darstellen und vor allem auch plotten – 280 Punkte horizontale Auflösung schafft er. Die Tastatur wurde auf den vollen ASCII-Zeichensatz erweitert, und außerdem stehen jetzt acht Funktionstasten (F1...F8) zum direkten Aufruf von Programmen zur Verfügung. Auch eine Reset- und eine NMI-Taste befinden sich auf der absetzbaren Tastatur. Eine rastende Taste „All Caps" dient zum Umschalten auf Nur-Großschreibung während des Programmierens, sie hat aber intelligenterweise keinen Einfluß auf die Wirksamkeit der Shift-Taste bei Ziffern- und Zeichentasten.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		[illegible]	[illegible]	⇒	↖	↗	↘	↙	←	→	↓	↑	↕	[illegible]	[illegible]	[illegible]
1	0	1	2	3	4	5	6	7	8	9	▼	▲	▶	◀	[illegible]	[illegible]
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	▒
8	α	β	γ	Δ	ε	δ	λ	μ	π	Π	σ	Σ	τ	Φ	Ψ	ω
9	Ω	°	∞	±	∓	÷	≈	≐	≠	≡	≤	≥	∪	∩	Γ	√
A	∫	[illegible]	§	¢	£	¥	‹	›	♠	♥	♣	♦	●	⌑	[illegible]	※
B	0	1	2	3	4	5	6	7	8	9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
C	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Ä	Å	ê	è	é	Ñ	Ö	ö
D	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	X	Ü	Ü	[illegible]	[illegible]	[illegible]	+
E	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	└	┘	┌	┐	[illegible]	├	┤	┬
F		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Der Zeichensatz des AIM-65/40, wie er vom eingebauten Selbsttest-Programm ausgegeben wird

Wem das Display nebst Drucker nicht ausreicht, dem stehen zwei RS-232-Schnittstellen zum Anschluß eines Terminals und eines größeren Druckers zur Verfügung; hierfür besitzt Systemsoftware auch besondere bildschirmorientierte Befehle. Das Kassetten-Interface schließlich arbeitet mit 1200 Baud sehr zuverlässig und ist mit jenem des AIM-65 voll kompatibel.
Der AIM-65/40 wird übrigens als offene Platine ohne Netzteil (5 V/3 A und 24 V/1 A) geliefert; um das Gehäuse muß man sich also selbst kümmern.

Jetzt dynamisches RAM

Zusätzlich zu den Monitorprogramm-ROMs kann der Anwender ROMs für die höheren Programmiersprachen Basic und Forth, für den 6502-Assembler oder auch für eigene EPROM-Programme auf die Platine stecken; die Fassungen eignen sich für 2-, 4- oder 8-KByte-EPROMs. Das standardmäßig mitgelieferte Monitorprogramm enthält einen komfortablen Texteditor, einen Line-by-line-Assembler für mnemonische Befehlseingabe und einen Disassembler; es liegt im Adressenbereich A000...BFFF. Außerdem enthält die Platine ein ROM mit I/O-Software im Bereich F000...FFFF. Ein ROM mit Mathematik-Routinen (E000...EFFF) ist auf Wunsch erhältlich, ebenso der Assembler (9000...9FFF). *Tabelle 1* zeigt die Speicheraufteilung.
Das beim AIM-65/40 nun dynamische RAM liegt im Bereich 0000...8FFF (36 KByte max.); normalerweise sind bei Lieferung jedoch „nur" 16 KByte RAM bestückt.
Die Adressen der I/O-Ports, der Timer und der Monitor-Unterprogramme sind ebenso wie die optionalen Zusatz-ROMs in keiner Weise mit dem AIM-65 kompatibel. Vorhandene ROMs z. B. für Basic lasen sich also für den AIM-65/40 leider nicht weiterverwenden. Ebenso müssen alle Maschinenprogramme erst an das neue Monitorprogramm angepaßt werden (*Tabelle 2*).

Drei CPUs in einem Rechner

Der AIM-65/40 arbeitet mit der bekannten CPU 6502 als Hauptprozessor; sie dient hauptsächlich zum Ausführen von Benutzerprogrammen, fragt aber über einen VIA-Baustein 6522 auch die ASCII-Tastatur ab, die (im Gegensatz zum AIM-65) wirklich alle ASCII-Zeichen enthält, wenn auch nicht die deutschen Sonderzeichen wie ä, ö, ü und ß.

Ein weiterer Prozessor, nämlich die vom mc-EMUF bekannte CPU 6504, dient ausschließlich zur Steuerung des Druckers (*Bild*), und ein zweiter 6504 bedient das 40stellige alphanumerische Fluoreszenz-Display.
Die drei Prozessoren sind nicht über Adressen- und Datenbus, sondern über eine ASCII-Schnittstelle miteinander verbunden, so daß sich Drucker, Display und Zentraleinheit prinzipiell auch getrennt verwenden ließen. Mit Steuerbefehlen läßt sich der Drucker aber auch im Grafik-Modus mit 280 Punkten horizontaler Auflösung betreiben. Alle Steuerzeichen entsprechen weitgehend der ASCII-Norm.

Dokumentation vorbildlich

Wie schon beim AIM-65, so ist auch beim AIM-65/40 die mitgelieferte Dokumentation in Form eines rund 7 cm dikken Bücherstapels wahrhaft vorbildlich, und dies ist sicher ein Argument für industrielle Anwendungen: Ein „User's Manual", das alle Gerätefunktionen, den Systemaufbau und die Bedienung beschreibt; ein Programmierhandbuch, das auf die Software der Prozessoren 6502 bzw. 6504 eingeht und – das ist ein besonderes Lob wert – zwei Büchlein mit den kompletten, kommentierten ROM-Assemblerlistings. Mitgeliefert werden auch ein 23seitiges „Summary Booklet", das in Kurzform über Monitor-, Editor- und Assemblerbefehle informiert sowie die wichtigsten Systemadressen enthält, ein Schaltbild und eine 6500-Programmierkarte mit allen CPU-Befehlen. Dazu kommen noch ausführliche Datenblätter der Bausteine 6522 (VIA), 6502/6504 (CPU), und 6551 (ACIA).
Derzeit steht die Dokumentation für den AIM-65/40 bis auf das Programmierhandbuch nur in englischer Sprache zur Verfügung; eine Übersetzung ins Deutsche ist aber bereits geplant.
Ein komfortabler Basic-Interpreter ist von GWK lieferbar; diese Firma stellte uns auch freundlicherweise das Testgerät zur Verfügung.

Tabelle 1: Speicheraufteilung beim AIM-65/40

Adresse	Bereich
FFFF–F000	I/O-ROM
F000–E000	Math-Pack-ROM (optional)
E000–C000	Basic-, PL-65- oder Forth-ROMs (optional)
C000–A000	Monitorprogramm- und Texteditor-ROMs
A000–9000	Assembler (optional)
9000–0800	Freies RAM
0800–04A0	Reserviert für externe Erweiterungen
04A0–0200	System-RAM
0200–0100	Stack (Page 1)
0100–0000	Page Zero (weitgehend frei)

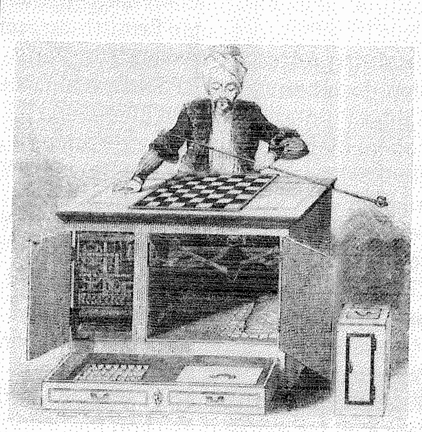
Tabelle 2: Einige Monitor-Unterprogramme

AIM-65	AIM-65/40	Name	Wirkung
E993	F233	INALL	Ein Zeichen vom aktiven Input Device einlesen
EA46	F3A4	NUMA	Ein Byte als zwei ASCII-Zeichen an das aktive Output Device senden
E9BC	F32B	OUTALL	Ein ASCII-Zeichen an das aktive Output Device ausgeben
E9F0	F38F	CRLOW	CR/LF an Display/Printer ausgeben

Für wen ist der Super-AIM gebaut?

Wie schon erwähnt, unterscheidet sich das Konzept des AIM-65/40 grundsätzlich von dem anderer Tischcomputer. Wer nur in Basic arbeiten möchte, sollte sich besser einen „herkömmlichen" Computer zulegen. Wer aber vorwiegend industrielle Steuerungen, Meßgeräte-Abfragen oder auch Einplatinen-Computer wie den mc-EMUF programmieren möchte, für den ist der AIM-65/40 mit Sicherheit besser geeignet als z. B. ein CBM mit seiner randvoll gefüllten „Zero Page" und nur acht I/O-Leitungen. Eine Stärke des AIM-65/40 ist sein superkomfortables Monitorprogramm und der sehr brauchbare Assembler – diese Dinge machen ihn prädestiniert für das Arbeiten in 6502-Maschinensprache. Nur der Laie hält momentan das 40stellige Display gegenüber einem Bildschirm für eine Beschränkung; die praktische Erfahrung zeigt schnell, daß es zusammen mit dem Thermodrucker völlig ausreicht. Fazit: Viel Leistung, leider nicht ganz billig.

Spruch des Monats



Eine Schachspielmaschine, die lernt, wird eine Vielfalt in ihrem Spiel zeigen, die von der Qualität der Spieler abhängt, gegen die man sie hat kämpfen lassen. Die beste Art, sie zu einer Meistermaschine zu machen, wäre wahrscheinlich die, sie gegen die allerverschiedensten guten Schachspieler einzusetzen. Auf der anderen Seite könnte eine wohlersonnene Maschine durch unüberlegte Wahl ihrer Gegner mehr oder weniger ruiniert werden. Auch ein Pferd wird durch die unüberlegte Wahl seiner Reiter zugrundegerichtet.

Norbert Wiener, 1949
(Norbert Wiener gilt als der Vater der Kybernetik)

Hans Georg Joepgen

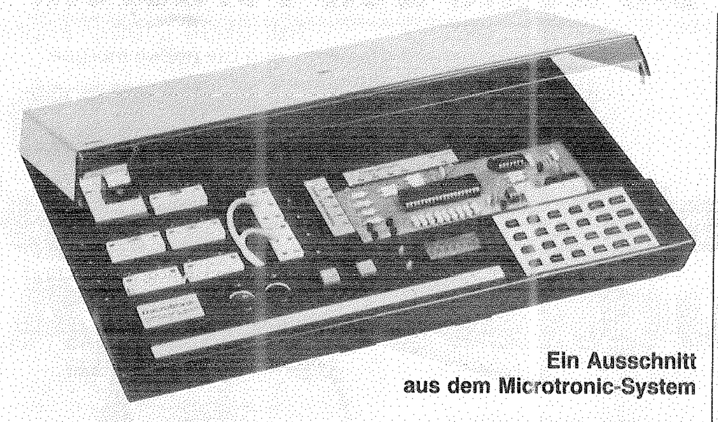
CP/M und neues Basic für Apple-Computer

Mit wenigen Ausnahmen basieren die Basic-Interpreter der gängigen Massen-Mikrocomputer auf Entwicklungen der amerikanischen Firma Microsoft. Nun hat das Haus Microsoft eine von Grund auf revidierte und mit traumhaft schönen Erweiterungen versehene Neuversion seines Interpreters vorgelegt, die wir praktisch erprobten. Microsoft-Basic 80, Version 5, ist seit jüngstem für die zahlreichen unter CP/M laufenden Computermarken erhältlich und gehört zum Lieferumfang des „Softcard-Paketes" für die Maschinen der Apple-Klasse.

Gleichgültig, ob man seine Basicprogramme bisher für eine Commodore- oder Tandy-Maschine schrieb, in Applesoft, Palsoft, Siemens-Basic oder DAI-Basic: Bei gewissen mathematischen Operationen wie beispielsweise der Division durch Null stieg der Interpreter aus, brach die Programm-Bearbeitung ab, druckte eine Fehlermitteilung und verabschiedete sich dann. Wollte der Programmierer derlei verhindern, so hatte er durch Vorprüfungen oder andere Mittel dafür Sorge zu tragen, daß die Fehlerbedingung nicht auftrat oder durch „ERROR TRAPPING" abgefangen wurde. Derlei Notwendigkeiten existieren im neuen Microsoft-Basic nicht länger: Der Rechner gibt einen Hinweis aus und fügt selbsttätig, je nach Fall, eine nahe bei Null liegende Ersatzzahl oder ein positives oder negatives „Rechner-Unendlich" ein; das Programm wird ohne Verzug fortgesetzt. Diese Annehmlichkeit, so augenfällig sie auch sein mag, ist aber nur eine von zahlreichen Verbesserungen. Mehr ins Gewicht fällt, daß nun Variablen-Namen mit bis zu 40 Zeichen Länge möglich sind, was das Erstellen selbstdokumentierender Programme ungemein erleichtert und Anfängern den Zugang zu Basic versüßt. Bereits in den für Tandy und DAI geschriebenen Basic-Dialekten fand sich die Variablen-Klasse „REAL-DOUBLE PRECISION", auf die PET-, Apple-, ITT-2020- und PC-100-Freunde verzichten mußten; sie steht in Basic 80 zur Verfügung. Bei der Definition von Anwenderfunktionen (DEFFN...) ist nun auch der Umgang mit Zeichenketten-Ausdrücken möglich, die Index-Untergrenze kann, wie bislang nur in Dialekten wie HP-Basic üblich, mit dem OPTION-BASE-Befehl zur Felddimensionierung manipuliert werden und der Textverarbeitung kommen mancherlei String-Operatoren zugute, die man sich in Microsoft-Basic alter Art durch umfängliche Hilfsroutinen selber zu machen hatte. Dazu: PRINT USING; bequeme Ermittlung von absoluten Adressen verwendeter Variablen durch einen VARPTR-Befehl („Variables Pointer"): Alles in allem eine in hohem Maße wünschenswerte und geglückte Fortentwicklung der Programmiersprache Nummer eins im Mikrocomputer-Bereich, wobei die Einführung eines neuen Statementpaares „WHILE" ... „WEND" (Iteration bis Abbruchbedingung zutrifft) strukturiertes Programmieren erleichtert und darüber hinaus noch bequem mit Diskettenaufzeichnungen umgegangen werden kann. Den Katalog der Vorzüge von Basic 80/5 beschließt die Möglichkeit, Programme vor unbefugtem Listen und Verändern höchst wirksam zu schützen (Betriebsart „PROTECTED").

Apple II und ITT 2020 werden zu Z-80-Computern durch Softcard

Bei unserer zwölfwöchigen Erprobung im praktischen Betrieb stand eine Enttäuschung am Anfang: Die „Softcard" mit ihrem Z-80-Mikroprozessor lief anfänglich nicht auf unserer ITT 2020. Nach Untersuchungen von Standard Elektrik Lorenz in Pforzheim war Ursache, daß die für den behäbigeren Apple geschaffene Zusatz-Karte mit der höheren Taktsignal-Frequenz der insofern flotteren ITT 2020 nicht zurechtkam. Einfügen eines Verzögerungsgliedes auf der Karte selbst, bestehend aus zwei monostabilen Multivibratoren 74121 im Werte von zusammen nicht mal drei Mark, beseitigte das Problem (Modifikation vorgeschlagen und durchgeführt von Theodor Kinkel). Einer gewissen Umgewöhnungsphase bedurfte es, bis wir mit dem an sich sehr leistungsfähigen Programm-Editor zurechtkamen, der zu Basic 80 gehört, und schließlich machten wir noch die Erfahrung, daß die Benutzung von numerischen Fließkomma-Variablen der Klasse „Doppelte Genauigkeit" (immerhin 16 signifikante Stellen!) die Arbeitsgeschwindigkeit des Interpreters dramatisch herabsetzte. Man wird diese Variablen nur dort benutzen, wo man sie wirklich braucht. Als sehr angenehm fanden wir schließlich noch, daß Integer-Konstanten unmittelbar in Oktal- oder Hexadezimalform und nicht nur in dezimaler Schreibweise notiert werden dürfen. Fazit: Nach Verfasser-Meinung ist allein das neue Basic den Preis für die Z-80-Karte (wir zahlten 280 Dollar einschließlich Luftfracht plus rund 15 Mark Zoll) allemal wert: CP/M und viele schöne Nutzprogramme einschließlich zweier hervorragend gemachter Handbücher, die zum Paket gehören, bekommt man – so betrachtet – „ dazugeschenkt".



Ein Ausschnitt aus dem Microtronic-System

Von Grund auf: Mikrocomputer-Lernsystem

Ohne Vorkenntnisse der Digitaltechnik vorauszusetzen und systematisch führt das Computersystem „Microtronic" in Hardware und Maschinensprache-Programmierung von Mikrocomputern ein. Um den Befehlssatz für den Anfänger überschaubar zu halten, wurde der 4-Bit-Einchip-Computer TMS 1600 von Texas Instruments gewählt. Der auf dem Mikrocomputer-Chip vorhandene Festwertspeicher (ROM) enthält einen kleinen Interpreter, der auch die Eingabe von Befehlen zuläßt, die nicht im ursprünglichen Befehlssatz des Prozessors vorhanden sind, z. B. Multiplikation und Division, und der auch für die Abfrage der hexadezimalen Tastatur und des achtstelligen LED-Siebensegment-Displays sorgt. Zur Demonstration sind auch zwei Programme schon fest gespeichert, nämlich ein Streichhölzchen-Spiel (das Nimm-Spiel) und die Funktion einer Digitaluhr. Fast noch wichtiger als diese am Lernen orientierte Hardware ist die systematisch aufgebaute und leicht verständlich geschriebene Anleitung, die aus zwei Bänden besteht. Sie beginnt mit dem Abruf der beiden Demonstrationsprogramme – zwei Erfolgserlebnisse, die einem schon einen ersten Eindruck von einem Computer geben. Anhand zahlreicher selbst einzugebender kleiner Programme werden die einzelnen Mikrocomputer-Befehle erläutert, aufgelockert von heiteren Illustrationen.

Der zweite Anleitungsband enthält in erster Linie zahlreiche Beispielprogramme: Spiele, Morse-Decoder, Simulation eines Taschenrechners, Datumsberechnung, Biorhythmus und sogar Sinus-Errechnung, Außerdem wird auf den Anschluß von Peripheriegeräten eingegangen, und dann werden auch solche Anwendungen wie Schaltuhr, Computer-Orgel, wie Reaktionszeit-Messung, Datenübertragung, Frequenzmessung oder Modelleisenbahn-Steuerung möglich. Da das Microtronic-System mit den Electronic-Studio-Baukästen des gleichen Herstellers hardwaremäßig kompatibel ist, sind Erweiterungen kein Problem.

„Microtronic" ist eine Neuentwicklung in der Reihe des „electronic-Experimentier-Systems". Es entstand in Zusammenarbeit mit unserem Elektronik-Magazin ELO und wurde bis vor kurzem unter dem Markennamen ELOtronic angeboten. Die verstärkte Zusammenarbeit mit der Herstellerfirma Busch & Co. führte zu der Namensänderung.

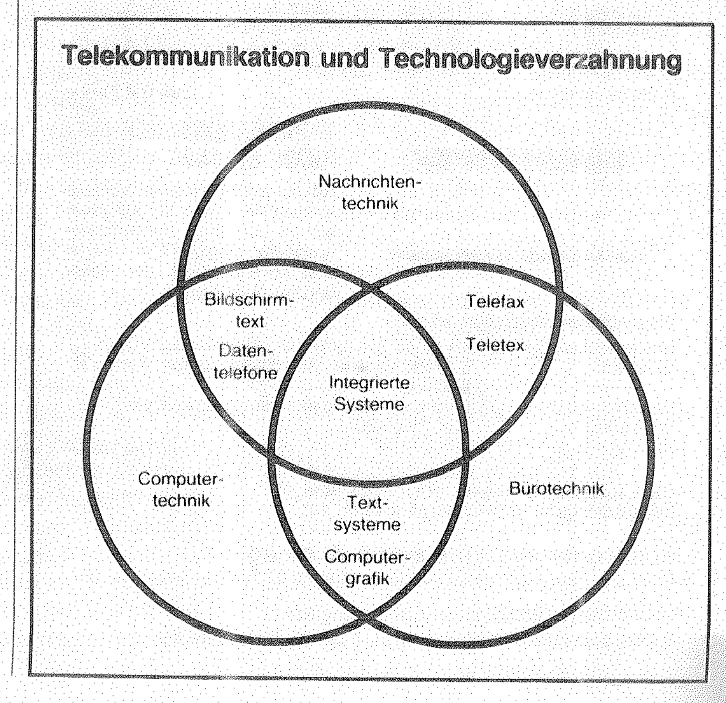
„Microtronic" erhalten Sie beim Elektronik-Fachhandel und größeren Buchhandlungen oder gegen Voreinzahlung direkt beim Franzis-Verlag (DM 379,– plus DM 3,– Porto/ Postscheckkonto München 813 75-809).

Systems 81, ein Riesen-Erfolg

Die Münchner Messe und Ausstellungsgesellschaft (MMG) war als Veranstalter der Systems natürlich verpflichtet, dieser Show der großen und kleinen Computersysteme und deren Anwendungen ein Motto mitzugeben. „In Systemen denken" lautete dieses Motto. Damit hat die Messegesellschaft Weitblick bewiesen, denn die Zukunft der Computer kann man heute nicht isoliert betrachten. Telekommunikation heißt das Schlüsselwort, das die Vernetzung der gegenwärtigen Kommunikations- und Datenverarbeitungssysteme beinhaltet. In Zukunft wird jede elektronische Büroschreibmaschine potentiell mit jedem Computer kommunizieren können. Oder jeder Telefonanschlußinhaber muß damit rechnen, daß ein Großrechner Daten bei ihm absetzen oder erfragen will. Integrated Services Digital Network (ISDN) ist der internationale Name für einen Dienst, der alle zur Nachrichten- und Datenverarbeitung fähigen Geräte und Systeme zu einem großen Supersystem zusammenfassen wird.

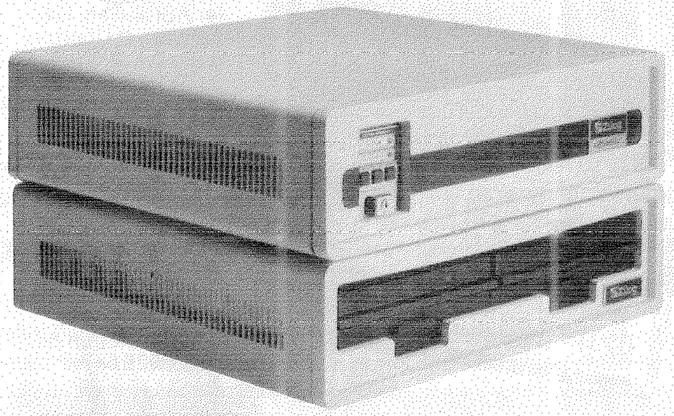
Für den Computermann und für den Mikrocomputermann bedeutet dies, daß beispielsweise die Nachrichtentechnik, die ja bei der Aufzucht der Computer teilweise Amme gespielt hat, heute wie damals von größter Wichtigkeit ist und bleibt. Und daß Anforderungen und Impulse bezüglich Hard- und Software verstärkt auch von außen an die Computerei herangetragen werden. Der begleitende Fachkongreß von Systems trug dem Motto weitgehend Rechnung. „Kommunikationssysteme als Produktionsfaktor", „Der Telekommunikationsmarkt der achtziger Jahre" und „Planungsraster für die Zukunft" lauteten zum Beispiel die Themen des Symphosiums A.

Daß dieses weittragende Motto der ganzen Messe eine Richtung geben konnte, das war das Verdienst der Messe- und Ausstellungsgesellschaft. Daß darunter eine lebendige und expansive Ausstellung mit wichtigen neuen Exponaten zustande kam, das war das Verdienst auch der Aussteller. mc schildert hier einige große und kleine Neuheiten, vor allem auf dem Mikrorechnerbereich, der sich ebenfalls auf den Weg macht, mit Riesenschritten die Zukunft zu erobern.



Was war los bei Zilog

Die Firma Zilog stellte zur Systems 81 einen repräsentativen Querschnitt ihrer leistungsfähigen Systeme einschließlich der zugehörigen Software aus. Neu ist das, auf Unix basierende 16-Bit-Mikrocomputer SYSTEM 8000, ein Mehrbenutzersystem. Multitasking mit ausgezeichneter Leistung. Das SYSTEM 8000 unterstützt bis zu 16 Benutzer. Die Z8001-CPU, unterstützt von 3mal Z8010 Memory-Managment-Unit, sorgt für hohen Datendurchsatz. Das Winchesterlaufwerk speichert 24 MByte und der Kontroller erlaubt den Anschluß von 4 Laufwerken. Für die Datensicherung ist eine Magnetbandkassette vorgesehen. Das SYSTEM 8000 ist in zwei Ausbaustufen verfügbar. Modell 20 hat 256 KByte, ECC-Speicher, eine 24-MByte-Winchester-Platte, eine 17-MByte-Magnetbandkassette und acht serielle Schnittstellen für die Bildschirme und ein paralleles Printer-Interface. Das Modell 30 beinhaltet alle Komponenten des Modelles 20 plus zusätzlicher 256 KByte Speicher und einem weiteren 24-MByte-Winchester-Laufwerk. Memory-Erweiterung bis zu 1,5 MByte ECC, Benutzer-Terminals bis zu 16 Benutzer und bis zu vier Winchester-Laufwerke können daran angeschlossen werden.



Das ist Zilogs Z-NET-Manager

Das Betriebssystem für das SYSTEM 8000 ist eine verbesserte Version von Unix und wird Zeus genannt. Für den Anwender stehen eine Reihe von Hilfsprogrammen zur Verfügung. Ein Mail-System erlaubt den Austausch von Informationen von Bildschirm zu Bildschirm. Ein programmiertes Wörterbuch überwacht die richtige Schreibweise von Wörtern. Die geschriebenen Texte können direkt in druckreifer Form ausgegeben werden. Sortier-Routinen und Rechner-Kopplungen sind verfügbar.
Anwender-Programme können in „C", Basic, Fortran oder Cobol geschrieben sein.
Ein weiteres interessantes Ausstellungsmodell war das Z-NET. Mit diesem lokalen Computer-Netzwerk können bis zu 255 Computer-Stationen oder Daten-Pooler zusammengeschlossen werden. Der Aufwand dafür ist minimal. Alle Stationen werden über ein einfaches Kabel angeschlossen (wie eine Antennenanlage in einem Hochhaus). Bei einer Erweiterung wird die zusätzliche Station mit einem T-Stück an das vorhandene Kabel angeschlossen. Die Übertragung von Rechner zu Rechner geschieht mit 90120 Bits pro Sekunde. Jede Station (Mikrocomputer) hat zwischen Rechner und Coax-Netz ein kleines Modem geschaltet, das den Datenverkehr überwacht.
Als Station ist die CPU-Box vom MCZ-2 vorgesehen. Dadurch hat der Benutzer seine eigene Z80A CPU, 64K-Byte-Speicher und, wenn benötigt, separate Floppy-Stationen. Für große Datenmengen wird eine Shared Data Station (SDS) eingesetzt. Die SDS besteht aus einer MCZ-2 CPU-Box und einer 10 MByte Plattenstation (5 Festplatten, 5 Wechselplatten).
Der intelligente Plattenkontroller bedient bis zu 40 MByte. Für den Anwender können Files „secret" abgelegt werden, d. h. sie sind den anderen Benutzern nicht zugänglich. Andere Möglichkeiten bietet ein I/O-Controller. Er kann als Terminal-Konzentrator, z. B. für Auftragseingang oder Datainput benutzt werden. Anschlüsse für Modem oder Plotter sind möglich. Alle Stationen können *einen* Drucker benutzen. Der Shared-Printer-Manager steuert und überwacht das Drucken.
Das Protokoll auf dem Z-NET ist eine Erweiterung des bewährten RIO-Betriebssystems. (Zilog, Zugspitzstr. 2a, 8011 Vaterstetten)



Modems – 1200 bis 9600 Bit/s

Als einer der weltweit führenden Modemhersteller wird Rockwell International nun auch in Europa durch die Aufnahme der Vertriebsaktivitäten dieser Produktgruppen Akzente setzen.
Als Einstand wird ein neu konzipiertes R24-Modem mit direkter Anschlußmöglichkeit an die Übertragungsleitung angeboten. Weiterhin wird ein Hochgeschwindigkeits-Modem, gut programmierbar von 4800 bis 9600 Bit/s als Komplettmodem, welches auf nur einer einzigen Doppel-Europa-Karte aufgebaut ist, angeboten werden. Beide Modems zeichnen sich durch außergewöhnlich hohe Intergrationsdichte und Modularität aus. Rockwell vertreibt auch die Computer AIM65 und AIM65/40 in Europa. (Rockwell International, Fraunhoferstraße 11, 8033 Martinsried)

Strichcode-Scanner mit Laser

Im Strichcode-Scanner der Firma Symbol Technologies International aus Brüssel ist ein He-Ne-Laser eingebaut. Das nur 680 g wiegende und recht kleine Gerät liest Strichcode zuverlässig aus einer Entfernung von 1 bis 8 cm. Es kann ohne weiteres durch Glas und transparente Hüllen oder von gewölbten oder weichen Oberflächen lesen. Laserscan 100, so heißt das Gerät, kann eine Lücke zwischen den Lesestiften und den großen stationären Scannern schließen. (Symbol Technologie International, rue Gachard 51, Bte 19-1050 Bruxelles)



Das ist „Laserscan 100" im Einsatz

Der Apple III kommt wirklich

Ab sofort will die Firma Apple die ersten Exemplare ihres mit Schmerzen in den USA geborenen Computers Apple III in Europa ausliefern. Originalton Apple: „Der neue Apple III ist das Ergebnis unserer Bemühungen, dem professionellen Benutzer von Personalcomputern Hardware- und Softwarelösungen an die Hand zu geben, die sich durch erhöhte Zuverlässigkeit und Leistungsfähigkeit auszeichnen. Frühere Probleme beim Apple III wurden durch Verbesserungen in unseren Beschaffungs- und Fertigungsmethoden, erweitertes und klar dokumentiertes Betriebssystem sowie durch leistungsfähige Applikationsprogramme von acht neuen bzw. verbesserten Softwarepaketen erzielt. Hierzu gehören: Das SOS-Betriebssystem (Sophisticated Operating System), das derzeit leistungsstärkste Betriebssystem, das auf Personalcomputer erhältlich ist. Die Programmiersprache Business Basic. Die Programmiersprache USCD-Pascal. VisiCalc III – eine erweiterte Version der bekannten „elektronischen Arbeitsblatt"-Software. Apple Writer III, ein komplettes Textverarbeitungspaket. Business Graphics III. Dieses Graphic-Paket erlaubt auch dem Nicht-Programmierer die Generierung von professionellen Linien-, Kreis- und Balkendiagrammen aus VisiCalc oder anderen Datenfiles. Access III. Dieses Datenkommunikationspaket ermöglicht den Anschluß des Apple III an einen großen Computer oder an weitere Apple-III-Systeme.



Ein großer Personal-Computer, der neue Apple III mit Profile

tionsprogramme vollständig beseitigt."

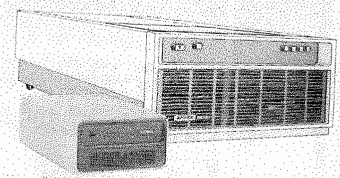
Der Apple III ist speziell für kleine Geschäftsbetriebe, professionelle Anwender und Manager konzipiert, deren Leistungsbedarf die Möglichkeiten des Apple II bereits überschritten hat.

Eine große Bandbreite der Apple-III-Einsatzmöglichkeiten wird durch die Verfügbarkeit Script III zum Formatieren und Ausdrucken von Pascal-Editor-Files.

Intern können maximal 256 KByte Speicher angesprochen werden. Mit vier Profile-Festplattenlaufwerken (im Bild zwischen Display und Grundgerät sichtbar) können bis zu 20 Millionen Byte on-line bereit gehalten werden. (Apple, Maximilianstraße 29, München 22)

Mini-Winchester, stoßfest

Ampex stellte ein neues 8-Zoll-Winchester-Laufwerk vor, das 80 oder 48 MBytes Kapazität zur Verfügung stellt. Diese Festplatte kann an beliebiger Stelle im Computersystem montiert werden. Es gibt eine Tischversion mit Gehäuse und eine Einbauversion ohne. Es läuft mit ±5 V, ±12 V oder 24 V Gleichstrom, die der Computer zu liefern hat. Besonders betont Ampex die Stoßfestigkeit des Laufwerkes, das – so sagt man – schon mal ganz aus der Hand fallen kann, ohne daß es beim Einsatz hinterher justiert werden muß. MTFB: 10 000 Stunden.



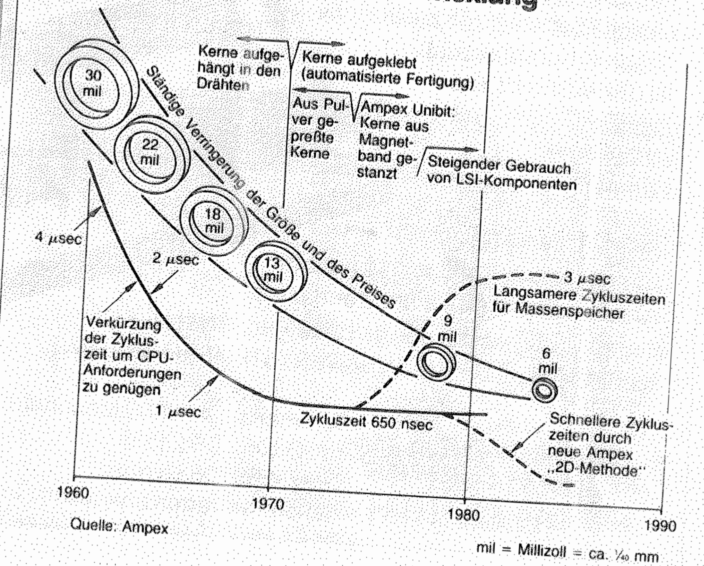
Viel Bit/cm³ speichert das neue Acht-Zoll-Laufwerk von Ampex (das Mini-Gerät vor Normal-Laufwerk)

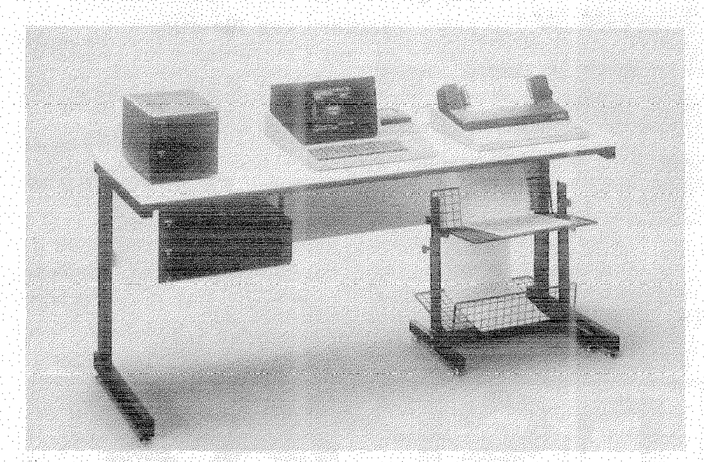
Kernspeicher contra Bubble-Memory

Nachdem die Technologie der Bubble-Memories auch nicht ganz ohne Probleme zu sein scheint, werden klassische Speichermedien wieder interessant. Ampex hat in den vergangenen Jahren die Technik der Kernspeicher weiterentwickelt und bietet jetzt Core-RAMs als nichtflüchtige Alternative an. Der Kernspeicher ist wesentlich schneller – er erreicht fast die Schnelligkeit von Halbleiterspeichern – und ist ungefähr gleich teuer, außerdem kann er als Arbeitsspeicher eingesetzt werden. 16 KByte werden auf einer 19-Zoll-Platine, 64 KByte auf zwei zusammenmontierten Boards untergebracht. Die Größe eines Magnetkerns beträgt teilweise nur noch 0,15 mm.

Auf der Systems wurden zwei Familien von Kernspeichern vorgestellt: RAMs als Arbeitsspeicher für Mikrocomputer, „Mikrostores" genannt, und Disk-kompatible Massenspeicher mit der Bezeichnung „Megastores". Die schnellsten Zugriffszeiten der Mikrostores liegen bei 326 ns. Die Kapazität der Megastore beträgt maximal 8 MByte. (Ampex Europa GmbH, Walter-Kolb-Str. 9, 6000 Frankfurt/M 70)

Kernspeicher Technische Entwicklung

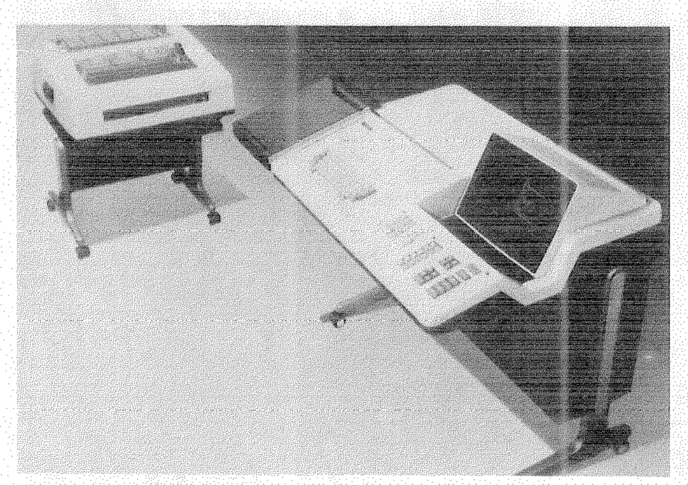




Ein Sharp MZ-80 K, systematisch ausgebaut

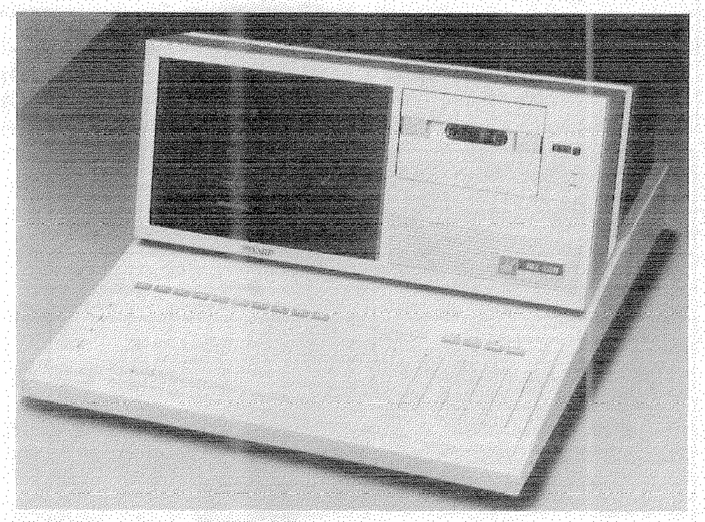
Sharp mit großer Palette

Für den bewährten Sharp MZ-80 K Personalcomputer wurde auf der Messe ein Katalog voller „deutscher" Software präsentiert. Vom Hotelwesen über Sanitär, Heizung und Klima bis zur Statistik reicht das Angebot an Software, die speziell für den deutschen Markt konzipiert ist.
Nach dem Basismodell MZ-80 K wird jetzt von Sharp ein weiteres auf der CPU Z80 aufbauendes Personalcomputersystem angeboten, der MZ-80 B. Er ist eine Generation weiter als andere Personalcomputer, denn er besitzt nur noch RAM-Speicherplätze. Er wird erst durch Booten zu einem Computer, wenn von dem eingebauten Kassettenlaufwerk her eine Sprache mit Betriebssystem geladen wird. Hervorzuheben ist sein graphisches Vermögen: Mit Anschluß eines Graphik-RAMs können 320 × 200 Bildpunkte einzeln angesprochen werden. Selbstverständlich können Floppy-Stationen und Drucker betrieben werden. Basicprogramme für den MZ-80 K sind übertragbar auf den MZ-80 B.
Sharps PC3201 ist ein System mit der Technologie und den Leistungsmerkmalen eines großen Computers – aber zum kleinen Preis. 32 KByte ROM und 64 KByte RAM sind eingebaut. Der freie Speicherplatz kann bis auf 112 KByte ausgeweitet werden. Basic ist die Programmiersprache. Finanzbuchhaltung, Lohn- und Gehaltsabrechnung sowie Auftragsabwicklung sind als Softwarepakete verfügbar. An branchenspezifischen Paketen wird gearbeitet.

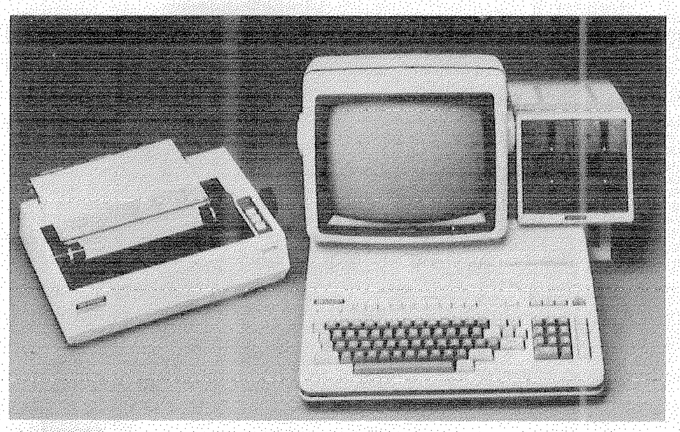


Obere Leistungsklasse: Sharp Hayac 3800

Das System Hayac 3800 ist für die gehobene kommerzielle Anwendung gedacht. Ab Anfang dieses Jahres können anspruchsvolle Anwender diesen kompakten Bürocomputer erwerben. Hayac-3800 spricht Cobol. Zwei Floppy-Disk-Laufwerke bieten 2 MByte Speicherkapazität. 10 MByte Festplattenspeicher kann man anschließen. Ein zweites Terminal erlaubt Zwei-User-Betrieb für größere Aufgaben.



Der Sharp MZ-80 B besticht durch modernes Design



Sharp PC-3201: Preiswerter Tischcomputer

Instant Pascal in ROMs

Für den AIM 65 von Rockwell International ist nun eine weitere Programmiersprache hinzugekommen. Das in 5 ROMs mit einer Gesamtspeicherkapazität von 20 KByte ausgelieferte „Instant Pascal" ist eine höhere Programmiersprache für hauptsächlich technisch-wissenschaftliche Anwendungen oder die Entwicklung von Compilierprogrammen. Der „P-Code-Interpreter/Compiler" von Rockwell wird aus Gründen optimaler Nutzung der verfügbaren Speicherkapazität sinnvollerweise auf einen 16-KByte-PROM-/ROM-Modul gesteckt (RM65-3216 (E)), wobei ein Pascal-ROM auf dem AIM 65 selbst verbleibt.
Die Programmiersprache Pascal gewinnt zunehmend an Bedeutung und ist hervorragend für die Beschreibung komplexer Algorithmen und auch für die Entwicklung anwendungsspezifischer Compilierprogramme geeignet. Der AIM 65 bleibt aktuell. (Rockwell International, Fraunhoferstraße 11, 8033 Martinsried)

AIM 65/40 Mikrocomputer.

Der große Bruder des erfolgreichen AIM 65 von Rockwell International

- Voll kompatibel mit der Rockwell Mikrocomputer-Board Familie RM 65
- IEEE 488-Bus-Controller
- 32 k Dynamic RAM Module
- 16 k PROM/ROM Module
- Single-Board-Computer
- 8 k Static RAM Module
- GPIO Module
- CRT-Controller
- ACIA Module
- insgesamt 21 verschiedene Module erhältlich

Als Mikrocomputersystem gibt Ihnen Rockwells neuer AIM 65/40 einen schnellen Einstieg in die Automatisierung von Laborsteuerungen, Meß- und Prüfvorrichtungen, Datensteuerungen und einer großen Anzahl von Industrieapplikationen.

Der AIM 65/40 verbindet 4 Module zu einem Mikrocomputersystem:

1. Keyboard mit 64 Tasten, 8 speziellen Funktionstasten
2. 40-stellige Anzeige mit separater Mikroprozessorsteuerung
3. 40-stelliger intelligenter Thermal Printer, graphikfähig und ebenfalls durch separaten Mikroprozessor gesteuert
4. Single-Board-Computer mit 6502 CPU, bis zu 80 K erweiterbares On-Board-Memory, extern bis 128 K, 3 VIA's und 7 Interrupts

Abgerundet wird dieses Hardwareangebot durch eine umfangreiche Softwareunterstützung wie

- interaktive Monitor
- Texteditor
- Firmware Optionen wie ASSEMBLER, BASIC, PL 65, FORTH

Und sollte dieses Angebot für Ihre Applikationen noch immer nicht ausreichen, steht Ihnen unsere vollkompatible Mikrocomputer-Board-Familie RM65 für alle möglichen Erweiterungen zur Verfügung. Selbstverständlich auch im Europakartenformat.

Alle Produkte erhalten Sie ab Lager.



SYSTEM-KONTAKT GmbH
Siemensstraße 5
7107 Bad Friedrichshall
Telefon: (0 71 36) 50 31
Telex 7 28 563

Vertrieb:

Deutschland:
SYSTEM-KONTAKT GMBH
Vertrieb
Neckarsulmer Straße 7
7100 Heilbronn
Telefon: (0 71 31) 7 10 94
Telex: 7 28 248

SYSTEM-KONTAKT GMBH
Niederlassung Berlin
Knesebeckstraße 49
1000 Berlin 15
Telefon: (0 30) 8 82 21 10
Telex: 185 305

Frankreich:
SYSTEM-CONTACT S.A.R.L.
4, Rue des Soers
F-67810 Holtzheim/France
Telefon: (88) 78 20 89
Telex: 890 266

SYSTEM-CONTACT S.A.R.L.
1, Place de la Balance – Silic 473
F-94613 Rungis-Cedex
Telefon: (1) 6 87 12 58
Telex: 202 312

USA:
SYSTEM-KONTAKT, INC.
6, Preston Court
Bedford/MA. 01730/USA
Telefon: 61 72 75-23 33 + 61 72 75-29 10
Telex: 7103 261 761